VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Prostředí pro emulaci síťových prvků z pohledu SNMP protokolu

Emulation Environment of Network Devices from Perspective of SNMP Protocol

2013 **Bc. Filip Bayer**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Filip Bayer**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Prostředí pro emulaci síťových prvků z pohledu SNMP protokolu
Emulation Environment of Network Devices from Perspective of SNMP Protocol

Zásady pro vypracování:

Cílem práce je vytvoření uživatelsky přívětivého nástroje, který umožní v prostředí jednoho počítače emulovat chování aktivních síťových prvků propojených podle zadané síťové topologie z hlediska protokolu SNMP. Vzniklé řešení umožní uživateli vhodným způsobem zadefinovat síťovou topologii, nastavit potřebné parametry jednotlivých aktivních prvků a spustit emulaci sítě. Práce bude řešena ve spolupráci s firmou SolarWinds a výsledný produkt bude zaměřen potřebám testování jejich dohledového řešení Orion Network Performance Monitor.

1. Seznamte se s problematikou protokolu SNMP a požadavky software Orion Network Performance Monitor.
2. Prozkoumejte stav existujících volně dostupných řešení, které se zabývají příbuznou oblastí.
3. Specifikujte požadavky na vyvíjený software a navrhněte způsob jeho implementace.
4. Navržené řešení implementujte a řádně otestujte.

Seznam doporučené odborné literatury:

MAURO, Douglas R.; SCHMIDT, Kevin James. Essential SNMP. 2, illustrated. USA : OReilly Media Inc, 2005. 442 s. ISBN 9780596008406.
STALLINGS, William. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. 3, illustrated. USA : Addison-Wesley, 1999. 619 s. ISBN 9780201485349.
WALSH, Larry. SNMP MIB handbook: essential guide to MIB development, use and diagnosis. illustrated. USA : Wyndham Press, 2008. 408 s. ISBN 9780981492209.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Martin Milata**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 15.7.2013 .....

Poděkování

Rád bych zde poděkoval Ing. Marku Žižlavskému a Ing. Davidu Pluháčkovi za odbornou pomoc při návrhu a realizaci implementace. Poděkovat bych chtěl Ing. Martinu Milatovi za vedení mé diplomové práce. Speciální poděkování patří také mým rodičům za podporu během celého mého studia.

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací aplikace pro emulaci síťových prvků v prostředí jednoho počítače, zapojených dle definované síťové topologie komunikujících pomocí protokolu SNMP. Aplikace je vytvořena a uzpůsobena pro testovací účely softwarového produktu Orion Network Performance Monitor firmy SolarWinds. Součástí aplikace je grafické uživatelské rozhraní umožňující nadefinovat jednotlivé síťové prvky a jejich vzájemné propojení, vygenerovat odpovídající data protokolu SNMP a spustit emulaci sítě. Nedílnou součástí je popis protokolu SNMP, zdrojů topologických dat a dalších teoretických východisek. Zmíněny jsou podobná existující řešení. Na závěr je provedeno testování a vyhodnocení výsledků.

Klíčová slova

SNMP, MIB, Responder, emulace zařízení, topologie sítě, .NET framework

Abstract

This thesis deals with the design and implementation of application used for emulation network devices in the environment of one computer, connected by defined network topology using SNMP. The application is created and designed for testing purposes for software product called Orion Network Performance Monitor of SolarWinds company. The application includes a graphical user interface to define the various network elements and their connections, generates corresponding SNMP data and has ability to start network emulation. An integral part is description of SNMP protocol, topological data and other theoretical knowledge. Mentioned are similar existing solutions. Finally, the testing is performed and results of testing are summarized.

Key words

SNMP, MIB, Responder, device emulation, network topology, .NET framework

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
API	Application Programming Interface	rozhraní pro programování aplikací
ARP	Address Resolution Protocol	protokol pro mapování síťových adres na fyzické adresy
CAM	Content Addressable Memory	paměť s mapováním fyzických adres na porty
CDP	Cisco Discovery Protocol	protokol pro sdílení informací mezi přímo připojenými sousedy v síti
DHCP	Dynamic Host Configuration Protocol	protokol pro automatickou konfiguraci počítače připojeného do sítě
DNS	Domain Name System	hierarchický systém doménových jmen
GUI	Graphical User Interface	grafické uživatelské rozhraní
IP	Internet Protocol	základní protokol síťové vrstvy přepravující data bez záruky
LLDP	Link Layer Discovery Protocol	protokol pro sdílení informací mezi přímo připojenými sousedy v síti
MIB	Management Information Base	databáze spravovaných informací o zařízení
NPM	Network Performance Monitor	softwarový produkt firmy Solarwinds určený k sledování sítě
OID	Object Identifier	jednoznačný identifikátor objektu z MIB
SNMP	Simple Network Management Protocol	protokol pro správu síťových zařízení
SQL	Structured Query Language	standardizovaný dotazovací jazyk pro práci s daty v relačních databázích

Obsah

1 Úvod

S rozvojem Internetu a neustálého připojování dalších zařízení do počítačových sítí je třeba řešit také problematiku spolehlivosti a údržby sítě. Výpadky jsou velice nepříjemné a mohou nás stát mnoho starostí i peněz. Abychom tyto problémy minimalizovali, jsou zde programy pro sledování sítě, které dokáží vzniklé chyby zjišťovat, udělat diagnostiku a umožnit změnu konfigurace. Zjišťovat výpadky můžeme mimo jiné taky z pohledu celé topologie, kterou lze na základě sběru informací o jednotlivých zařízeních zrekonstruovat. Dovíme se, jak vypadá celá aktuální logická struktura síťové topologie včetně rozmístění jednotlivých prvků vizualizovaných na síťové mapě. V případě problému pak víme, kde zasáhnout.

Cílem mé diplomové práce je vytvoření uživatelsky přívětivého programového nástroje pro emulaci síťových prvků propojených podle zadané síťové topologie komunikující přes protokol SNMP. Uživatel bude schopen vhodným způsobem pomocí grafického uživatelského rozhraní zadefinovat síťovou topologii, nastavit potřebné parametry jednotlivých aktivních prvků a spustit emulaci sítě. Následně pak budou spuštěny procesy zjišťování topologie sítě softwarového produktu Orion Network Performance Monitor firmy SolarWinds, se kterou na své práci spolupracuji. Výsledky zjištění topologie sítě Orionem pak budou porovnány s původní emulovanou topologií zadanou uživatelem a vyhodnoceny. Celý program poslouží jako řešení pro účely testování.

V první části textu se zaměřím na teoretický základ, ze kterého vycházím. Jedná se o popis protokolu SNMP a s ním souvisejících pojmů. Následně krátce vysvětlím, jak lze pohlížet na celou problematiku monitoringu a jak fungují aplikace pro sledování sítě, zejména softwarový produkt Orion firmy SolarWinds. Dále zmíním způsoby, jak se emuluje síť a její topologie. Podívám se na existující volně dostupná řešení a zhodnotím aktuální stav. V druhé části seznámím čtenáře s návrhem vlastní aplikace se všemi využitými MIB včetně seznamu konkrétních OID. Postup implementace celé aplikace, zvolené prostředí a ukázky kódu jsou součástí třetí části textu. Ve čtvrté části poté vyhodnotím, jak mi celá aplikace fungovala, porovnám výsledky a vyvodím závěry.

2 Teoretická část

2.1 Protokol SNMP

Simple Network Management Protocol je standardizovaný protokol aplikační vrstvy sloužící pro monitorování a správu síťových zařízení v IP sítích. Je podporován celou řadou zařízení, jako jsou aktivní síťové prvky, servery, tiskárny, různá čidla apod. Definuje jednoduché komunikační schéma pro přenos informací týkajících se stavu síťových prvků a provozu na nich. Součástí standardu je i definice databázové struktury (MIB) a datových objektů (OID), do nichž se data ukládají.

SNMP protokol tímto tvoří užitečný nástroj pro hledání a řešení problémů na síti, nepřímo umožňuje zvýšení výkonu sítě či další plánování růstu sítě. Většina prostředků a nástrojů pro správu sítě tohoto protokolu využívají.

2.1.1 Historie a vývoj verzí

Počátky vzniku protokolu SNMP se datují od roku 1988, kdy se začala vyvíjet jeho první verze (SNMPv1). Její slabinou je ovšem nedostatečné zabezpečení. To funguje tak, že přístup je umožněn na základě tzv. Community stringu, což je nešifrovaný řetězec s heslem. Pomocí tohoto řetězce se ve většině implementacích určuje také úroveň oprávnění. Pro čtení se používá obvykle řetězec „public“ a k zápisu řetězec „private“. Nicméně použít můžeme řetězec libovolný.

V roce 1993 přichází návrhy řešení druhé verze (SNMPv2), která má vylepšit nedostatky verze první. Vylepšení se týkají zvýšené úrovně bezpečnosti, schopnosti požadavku většího množství dat (bulk retrieval), možnosti komunikace správce-správce a několika dalších funkcí. Návrh řešení bezpečnosti se však ukázal jako příliš složitý a nebyl široce přijat. Zůstalo se u Community stringu a otázka bezpečnosti se odložila do třetí verze (SNMPv3). Ta byla standardizována v roce 2004 a konečně přinesla podporu pro bezpečnou autentizaci a šifrování pomocí algoritmů DES/AES.

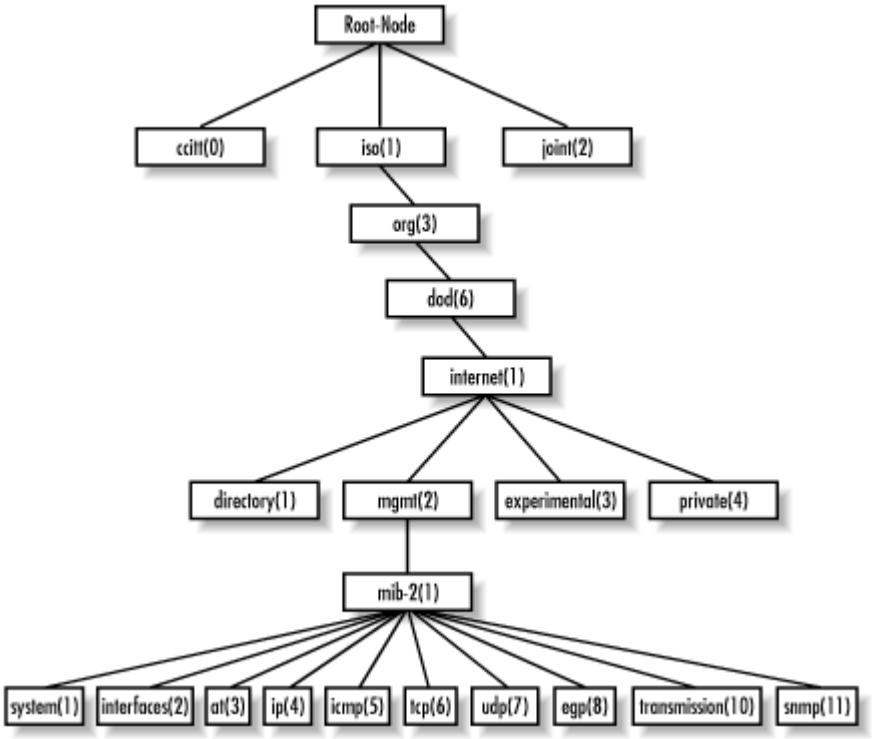
2.1.2 Princip fungování

Protokol vyžaduje pro komunikaci dvě strany. Na straně monitorované je spuštěn tzv. agent a na straně monitorovací správce (manager). Správce posílá agentovi dotazy, ten je vyhodnocuje a posílá zpět odpovědi. Hodnoty může získávat více správců a mohou se dotazovat kdykoliv nezávisle na sobě. Krom čtení umožňuje protokol správci i nastavení některých sledovaných hodnot na zařízení. Agent je schopen zasílat odpovědi také automaticky, typicky když nastane nějaká výjimečná situace (např. přehřívání, překročení limitních údajů apod.). Tento typ zpráv se označuje jako „trap“.

K přenosu dat je využito nespojovaného UDP protokolu. Standardně se používá port 161 (SNMP) na straně agenta (pro dotazy) a port 162 (SNMPTRAP) na straně serveru (pro trapy).

2.1.3 MIB databáze

Management Information Base (MIB) je hierarchická, stromově orientovaná definice databázových vlastností všech objektů, které mohou být spravovány. Jsou zde specifikovány informace, které musí jednotlivé typy síťových prvků udržovat a zpřístupňovat správcům. Každý objekt v MIB je jednoznačně identifikován pomocí číselného označení OID – Object Identifier. Tento identifikátor je tvořen sekvencí dekadických číslic, oddělených tečkou. Může reprezentovat jednotlivé MIB větve nebo konkrétní objekty v rámci MIB. Pro každou číselnou hodnotu je definováno také textové označení.



Obrázek 2.1: Ukázka struktury MIB

Příkladem OID může být třeba hodnota 1.3.6.1.2.1.2.2.1.1.1, která v textové reprezentaci odpovídá označení iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifIndex.1.

2.1.4 SNMP operace

SNMP protokol umožňuje několik základních operací mezi agentem a správci.

- **GetRequest**

  Požadavek správce o získání informace od agenta. Informací se rozumí hodnota konkrétního OID obsaženého v MIB tabulce agenta.

- **SetRequest**

  Požadavek správce agentovi o nastavení určité hodnoty konkrétního OID.

- **GetNextRequest**

  Požadavek správce o získání hodnoty OID, která následuje po hodnotě OID v požadavku.

- **GetBulkRequest**

  Požadavek správce agentovi o získání více OID hodnot najednou.

- **Response**

  Potvrzovací odpověď agenta na výše uvedené požadavky. Může obsahovat také informaci o chybě, pokud nastane.

- **Trap**

  Asynchronní zpráva od agenta správci, jako reakce na určitou událost. Například po překročení mezních hodnot apod. Zpráva je zaslaná nad nepotvrzovaným UDP, tudíž není jistota, že byla doručena.

- **InformRequest**

  Asynchronní zpráva od agenta správci nebo mezi správci doplněná o potvrzení, které je odesláno po doručení zprávy.

Vypracování této podkapitoly vychází z [1], [2], [3], [4], [5] uvedených v seznamu použité literatury.

2.2 Aplikace pro sledování sítě

Pokud se máme bavit o sledování sítě, je třeba se nejprve zorientovat v možnostech jak na celou problematiku nahlížet. Je zřejmé, že po aplikacích tohoto druhu budeme chtít minimálně 2 věci:

1) Informace o aktuálním stavu sledovaných položek (např. vytížení serveru, obsazenosti portů přepínače, připojených zařízeních) včetně historických záznamů pro případné plánování dalšího využití.

2) Upozornění, že někde došlo k problému (zařízení není dostupné, překročení mezních hodnot apod.).

Toto jsou základní kritéria, která by měly splňovat aplikace pro sledování. S tímto úzce souvisí fakt, že když už o problému víme, potřebujeme na něj rychle a pružně reagovat např. změnou konfigurace na sledovaných zařízeních. Těmito funkcemi však disponují aplikace pro správu. V praxi se zmíněné oblasti mohou často spojovat či různě kombinovat. Výsledkem je pak komplexní software umožňující jak sledování, tak správu.

2.2.1 Rozdělení aplikací pro sledování sítě

2.2.1.1 Pole oblasti sledování

Rozhodujícím aspektem výběru monitorovacího systému bude zajisté oblast, kterou budeme chtít sledovat. Nabízí se poměrně velké množství možností, které si stručně rozebereme a popíšeme.

Sledování serverů a aplikací

Tato skupina programů bude důležitá především pro administrátory zodpovědné za údržbu fyzických a virtuální serverů nebo osoby starající se o optimální běh a dostupnost důležitých aplikací.

Sledovat lze hardware fyzických serverů, kam patří informace např. kolem vytížení procesoru, operační paměti, využití pevných disků, stav chlazení.

Dále můžeme sledovat operační systém serveru. Zde bych zmínil běžící služby a procesy, aktuální využití operační paměti, virtuální paměti, stránkovacího souboru, stav logických jednotek, diskových operací apod.

Nebo můžeme sledovat programy běžící pod operačním systémem jako např. emailový server, databázový server, webový server, adresářový server či přímo konkrétní webové aplikace.

Pro monitorování informací z operačních systémů Microsoft Windows používáme protokolu WMI.

Sledování sítě

Oblast sítí a síťové infrastruktury umožňující komunikaci mezi zařízeními je jednou z oblastí, kde je sledování velice žádoucí a používané. Síťoví specialisté a administrátoři spravující sítě potřebují sledovat jednotlivá zařízení připojená na síti (směrovače, přepínače, servery, firewally, wifi zařízení apod.).

Zajímá nás stav zařízení, jeho dostupnost a výkonnost (např. ztrátovost paketů, doba odezvy, vytížení CPU, paměti). Dále jednotlivá síťová rozhraní, kde sledujeme mimo stavu a dostupnosti také aktivní spoje tj. zdali jsou připojena k nějakému dalšímu rozhraní jiného zařízení.

Odtud se dostáváme k oblasti, se kterou značně souvisí i moje diplomová práce a tou je tzv. Síťová topologie. Což je zachycení reálné struktury vzájemného propojení jednotlivých zařízení mezi sebou. Rozmístění jednotlivých uzlů v síti se zobrazují pro názornost zpravidla do map.

Další velkou oblastí, kterou lze v síti sledovat je IP adresovací prostor. Zde jsme schopni sledovat DHCP a DNS servery. Získáme přehled o obsazenosti jednotlivých adresových rozsahů v podsítích a stav jednotlivých IP adres.

Jako poslední oblast z mnoha dalších zmíním také sledování datových toků na síti (tzv. NetFlow). Zde můžeme zjistit, jakým způsobem jsou využívány připojené linky. Máme zde informace o protokolech, aplikacích využívající šířku pásma, víme od jakého zdroje ke kterému data putují, jsme schopni identifikovat místa generující nejvyšší zátěž a také rozpoznat na kterých službách kdo tráví nejvíce času.

Pro sledování sítě lze dobře využít protokolu SNMP. K sledování datových toků nejčastěji protokolu NetFlow.

Bezpečnost a logování

Sledování oblasti logování funguje na principu sběru logovacích dat (např. syslogy apod.) a dat ohledně událostí z nejrůznějších zdrojů a následném převodu těchto zdánlivě nepřehledných informací to uživatelsky srozumitelné formy. Těmito zdroji mohou být např. aplikace, servery, databáze, sítová zařízení, firewally, virtuální stroje apod. Tyto programy data přehledně rozdělí a umožní uživateli sledovat, co se na kterém zdroji dat odehrává. Provádí analýzu dat a na základě této analýzy upozorní uživatele na nepovolenou činnost či cokoliv co by odpovídalo určité hrozbě či možnému nebezpeční pro systém. Umožňují uživateli nadefinovat vlastní logiku kontroly a reakcí na určité události. Příkladem software pro tyto účely je Log & Event Manager od firmy SolarWinds.

Datové uložiště

Pomocí programů pro sledování datových uložišť jsme schopni udržet pod kontrolou kompletní datovou infrastrukturu zahrnující jak zařízení připojená k serverům přímo (DAS), tak uložiště připojená k místní síti (NAS). Nechybí podpora ani pro vysokorychlostní optické sítě ve kterých je připojeno větší množství vzájemně propojených úložných prostorů (SAN), které jsou odděleny od komunikační sítě podniku. Jsme tak schopni velice efektivně spravovat tento velký prostor a mít k dispozici kompletní informace o využití uložišť, což lze použít i pro budoucí rozšiřování a plánování.

Virtualizace

Poslední oblastí pro sledování, kterou zde zmíním je oblast virtuální infrastruktury. Máme zde celkový náhled na veškeré úrovně virtualizace. Pohled na clustery, datová uložiště, virtuální stroje a dokonce i aplikace, umožňující odhalit aktuální problémy, ale i problémy v minulosti. Podobně jako u sledování serverů můžeme u virtuálních strojů sledovat vytížení procesoru, paměti, apod. Jsme schopni identifikovat nevyužité virtuální stroje, stav snapshotů. Uzpůsobit si celkové další využití.

2.2.1.2 Podle zvolené technologie

Máme-li jasno o oblasti monitorování, je třeba rozhodnout, jakou technologii zvolit pro přístup k informacím. Na výběr jsou v zásadě dva přístupy:

- Monitorování s agentem
- Monitorování bez agenta

Monitorování s agentem

Na monitorovaný systém, odkud chceme sbírat informace, musíme nainstalovat další program – agent, který se stará o zpřístupnění informací. Výhodou tohoto řešení je velká míra přizpůsobitelnosti a rozšiřitelnosti agenta pro konkrétní potřeby. Máme k dispozici detailní monitorovací informace, protože agenti jsou vyvíjeni přímo na míru konkrétním systémům. Monitoring je robustní. Na druhou stranu mohou nastat problémy s nasazením agenta do systému a jeho následnou údržbou. To vyžaduje více úsilí a přijde nás to tím pádem dráž.

Výjimku tvoří standardizovaní agenti, se kterými se setkáme typicky například u protokolu SNMP. Zde jsou charakteristiky podobné jako u monitorování bez agenta (viz. Monitorování bez agenta).

Monitorování bez agenta

V případě monitorování bez agenta není nutné instalovat žádný další program. O sběr informací se starají standardizované protokoly, které jsou pro sledování určené (NetFlow, IDS). Tyto protokoly musejí být na straně sledovaného zařízení podporovány a nakonfigurovány. Výhodou tohoto přístupu je snadné zprovoznění a údržba monitorování. Jako nevýhodu by se dalo považovat jisté omezení v množství zařízeních podporující monitorací protokoly a možnost sledovat pouze určité množství informací.

2.2.2 Orion Network Performance Monitor

Orion Network Performance Monitor firmy SolarWinds je komerční produkt určený pro oblast sledování sítí. Jedná se o řešení využívající technologii monitorování s agentem. Pro sledování využívá především protokolu SNMP, případně WMI. Aplikace funguje na webu a pro ukládání dat využívá databázi SQL.

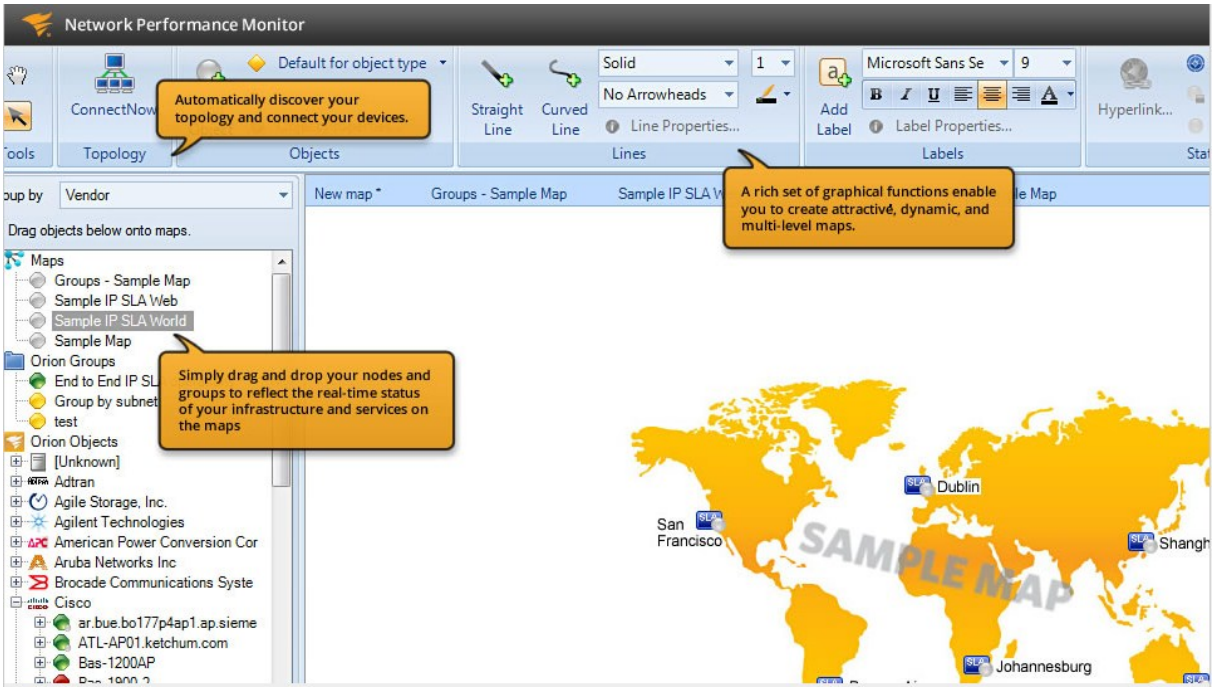
Webové rozhraní (Obrázek 2.2) poskytuje množství informací pro rychlé zjištění, diagnostiku a řešení problémů s výkonem v síti. Zobrazuje stav a dostupnost zařízení v reálném čase a pomocí grafické reprezentace umožňuje na první pohled určit kde je problém. Důraz je kladen na jednoduchost zprovoznění, kterou zvládne i méně zkušený IT administrátor.

Obrázek 2.2: Webové rozhraní aplikace Orion Network Performance Monitor

Data o výkonu sítě mohou být zobrazena v přizpůsobitelných grafech, tabulkách, mapách či top 10 seznamech. Funkce zjišťování sítě automaticky objevuje přepínače, směrovače, firewally,

bezdrátová zařízení, servery nebo jakákoliv další zařízení podporující SNMP. Hlavní přehledová stránka zobrazuje množství různě zaměřených informačních boxů. Box pro uzly zobrazuje seznam právě monitorovaných uzlů v síti spolu s jejich stavem seskupené dle různých kritérií (např. podle výrobce, typu stroje, umístění atd.).

Network Atlas s funkcí ConnectNow umožňuje automaticky vytvořit a přizpůsobit síťovou mapu a vizualizovat tak vaši síť – síťovou topologii. Má propracovaný mechanizmus výpočtu síťové topologie podle sběru topologických dat. Výsledky výpočtu jsou udržovány ve SQL databázi v odpovídající tabulce, ve které jsou informace ohledně jednotlivých spojení. Přesněji tedy o tom, které uzly jsou navzájem spojeny a na jakých rozhraních. Čísla indexů odkazují do tabulek s kompletními seznamy uzlů a rozhraní, kde jsou další podrobnější informace.



Obrázek 2.3: Aplikace Network Atlas

Topologická data jsou uložena zvlášť v tabulkách odpovídajících danému typu topologických dat (např. CDP, LLDP, ARP). Jednotlivé MAC adresy přiřazené k uzlům jsou také uloženy zvlášť v další tabulce. Jelikož topologická data budu využívat také ve své aplikaci, věnuji této oblasti zvláštní kapitolu, kde vše proberu a vysvětlím.

2.3 Emulace sítě a její topologie

Nejprve si řekněme, co to vlastně emulace sítě je. Síťový emulátor je software, který umožnuje simulovat chování zařízení v síti. Pracuje s fyzickou sítí a je schopen upravit chování zařízení různými způsoby jako např. měnit rychlost linky, generování zpoždění, ztrátovost paketů, prohazovat pořadí paketů a mnoho dalších. Využívá síťových protokolů ke generování dat a na venek se tváří jako reálné zařízení, které je na první pohled nerozpoznatelné od těch fyzických.

Síťové emulátory lze rozdělit podle úrovně emulace na 2 typy:

- Emulátory hardwarové vrstvy zařízení
- Emulátory síťových vrstev – protokolů

Emulátory hardwarové vrstvy zařízení

Tyto emulátory se snaží emulovat chování procesoru a vnitřního hardware síťového zařízení – typicky směrovačů, ale i dalších prvků sítě. Výhodou je velká míra reálnosti chování zařízení, avšak za cenu vyšší výpočetní složitosti.

Emulátory síťových vrstev zařízení - protokolů

Zde jde o emulaci konkrétních síťových protokolů, např. komunikace přes SNMP, rodiny protokolů TCP/IP nebo směrovacích protokolů jako například BGP, OSPF.

Z pohledu generování dat lze emulaci rozdělit:

- Statická emulace – data se vygenerují jednou, pak už se nemění
- Živá emulace – zařízení neustále generuje nová data a snaží se napodobit chování v reálném čase

Můj program bude spadat do kategorie Emulátorů síťových vrstev se statickou emulací.

2.3.1 Existující volně dostupná řešení

Existující volně dostupná řešení zabývající se problematikou emulace sítě z pohledu SNMP protokolu jsem našel pouze dvě – SNMP Simulator (SourceForge) a SNMP Agent Simulator (Verax Systems). Oba tyto nástroje však umožňují pouze emulaci odpovědí SNMP agenta na základě existujícího podkladu (např. snmpwalk) definující zařízení. Vlastní SNMP data vygenerovat nedokáží. Přesně s naší aplikací se tedy srovnávat nedají, protože naše aplikace nad tímto staví vlastní funkcionalitu zaměřenou na definici sítě, jejich parametrů a vygenerování SNMP dat, odpovídajících nadefinované struktuře sítě. Zmíněné nástroje už pak těchto dat využívají k emulaci jednotlivých zařízení.

SNMP Simulator

Emulátor zařízení naprogramovaný v Pythonu. Dostupný pod projektem SourceForge. Vyznačuje se těmito vlastnostmi:

- Čistě Python aplikace, jednoduše nastavitelná a přenosná
- Podpora SNMPv1/v2c/v3
- USM podporuje MD5/SHA autentifikaci a DES/AES/3DES/AES256 zabezpečení
- Může operovat přes IPv4 a/nebo IPv6 transporty
- Odpovědi se mohou lišit podle SNMP Community, Context, zdrojových/cílových adres a portů
- Schopnost sbírat a ukládat snapshoty SNMP agentů pro pozdější emulaci
- Podpora spuštění emulace na základě MIB souborů, snmpwalků a sapwalk výstupů
- Sbírá data pro emulaci z externích programů nebo SQL databáze
- Podpora spuštění SNMP TRAP/INFORMs na SET operacích
- Schopnost zároveň emulovat desítky až stovky agentů
- Snadné rozšíření pomocí Python skriptování
- Dostupnost také jako samostatně spustitelná Windows aplikace

Tento emulátor je zajímavý díky možnostem zabezpečení. Krom podpory autentizace a šifrování, je schopen pracovat s USM (User Security Model). Jedná se v podstatě o zabezpečení pomocí definování uživatelů s různými možnostmi zabezpečení a různým přístupem k datům v závislosti na tomto zabezpečení. Je schopen napodobit také živou emulaci, kdy na základě definovaných parametrů generuje pro některé údaje měnící se data.

SNMP Agent Simulátor

Emulátor zařízení, vyvíjený společností Verax Systems. Dokáže simulovat SNMPv1/v2c agenty na jednom hostitelském počítači na standardním portu 161 pomocí tzv. multinettingu. Vyznačuje se těmito vlastnostmi:

- Podpora mnoha agentů ve více sítích na jednom hostiteli
- Konfigurační soubory pro SNMP odpovědi jsou kompatibilní se SNMP walk výstupem pro snadné vytvoření simulací s daty ze stávajících zařízení
- Přes 20 předkonfigurovaných souborů odpovídajících simulaci zařízení Windows, Linux, Cisco, Juniper a dalších
- Sada pravidel pro modifikací odpovědí agenta obsahující:

- Generování náhodných MAC a IP adres, včetně pokročilých scénářů jako náhodnost pouze částí adres apod.
- Podpora celočíselných aritmetických operací (např. návratová hodnota na základě součtu dvou dalších hodnota apod.)

Program je napsán v Javě a je tedy určen jak pro Linuxové distribuce (SuSE, RedHat Enterprise, Debian), tak pro Windows systémy.

2.4 Zdroje topologických dat

Jelikož budeme v naší aplikaci pracovat se síťovou topologií, proberme si, odkud můžeme tyto data brát či jakých dalších zdrojů a protokolů můžeme využít k tomu, abychom na základě potřebných dat mohli zrekonstruovat topologii sítě. Probere si zde především protokoly 2. vrstvy ISO/OSI modelu, protože jiných, krom jedné výjimky, využívat nebudeme.

2.4.1 Cisco Discovery Protocol

Cisco Discovery Protocol je proprietární protokol společnosti Cisco, který je určen pro sdílení informací mezi přímo připojenými sousedy v síti. Téměř všechny zařízení Cisco tento protokol podporují. Funguje nad 2. vrstvou ISO/OSI modelu (Linková vrstva) a není závislý na použití dalších protokolů ani způsobu fyzického zapojení. Síťové rozhraní však musí podporovat SNAP (Subnetwork Access Protocol), což většina používaných přenosových technologií splňuje.

Obě komunikující strany posílají v pravidelných intervalech stavové zprávy, ve kterých souseda informují o svém stavu a stavu svých rozhraní (včetně přiřazených adres na rozhraních, základní identifikace systému i virtuálních sítí). Tyto informace jsou posílány jako multicast a také ukládány do tabulek, kde jsou pak přístupné přes SNMP v CISCO-CDP-MIB.

Každé zařízení si tedy uchovává vlastní CDP cache, která může vypadat například takto:

```
Router_2#show cdp neighbors

Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID     Local Intrfce     Holdtme    Capability  Platform  Port ID
Router3          Ser 1          120           R        2500      Ser 0
Router1          Eth 1          180           R        2500      Eth 0
Switch1          Eth 0          240           S        1900      2
```

Zajímat nás bude především identifikátor sousedícího uzlu (Device ID), rozhraní přes které jsme k uzlu připojeni (Local Intrfce) a vzdálené rozhraní připojeného sousedícího uzlu (Port ID). Při

detailnějším zobrazení pomocí příkazu `show cdp neighbors detail`, získáme krom dalších informací také IP adresy, případně MAC adresy sousedících uzlů.

2.4.2 Link Layer Discovery Protocol

Link Layer Discovery Protocol (LLDP) je standardizovaný protokol (není tak závislý na specifickém výrobci a představuje tak vhodný prostředek do prostředí zařízení různých výrobců) a podobně jako Cisco Discovery Protocol používá ke komunikaci linkovou vrstvu. Standard je spravován Institutem pro elektrotechnické a elektronické inženýrství (IEEE) pod označením 802.1ab. Definuje způsob, jakým síťová zařízení v prostředí Ethernetu mohou informovat své sousedy o vlastním nastavení, a umožňuje samotnému zařízení přistupovat k informacím z vyšších vrstev ISO/OSI modelu.

Protokol funguje nad specifickou multicast adresou v nespojovaném, jednosměrném a pouze informace propagujícím režimu. Sousední uzly tak nemohou ostatní o data požádat, ale čekají na pravidelné oznámení. V důsledku toho LLDP neřeší monitorování změn stavů mezi uzly.

Vylepšení LLDP-MED (Media Endpoint Discovery) umožňuje automatické zjištění síťových politik (virtuální lokální sítě a klasifikaci síťového provozu z pohledu kvality služeb) a přináší vylepšenou správu technologie VoIP stanic.

Většina protokolových dat je adresována v LLDP-MIB a je tak přístupná přes SNMP.

2.4.3 Content Addressable Memory tabulka

Jedná se o data z prostředí síťových přepínačů. Ty pracují na druhé vrstvě, takže přistupují k MAC adresám a podle nich rozesílají provoz. Když přijde nějaký rámec, tak se přečte zdrojová MAC adresa a v paměti se sestavuje tabulka portů, na kterém je uzel s vybranou MAC adresou připojen. Portem rozumíme rozhraní, po kterém rámce do přepínače přišly. Pokud později přijde rámec adresovaný pro uzel, jehož záznam je v CAM tabulce obsažen, je odeslán z přepínače přes uložené rozhraní. Pokud záznam v CAM tabulce uveden není, rámec se pošle na všechna rozhraní kromě vstupního.

Data z CAM tabulky jsou přístupná přes SNMP v tabulce BRIDGE-MIB a Q-BRIDGE-MIB. Konkrétně se jedná o tabulky dot1dTpFdbTable a dot1qTpFdbTable.

2.4.4 Address Resolution Protocol

Dalším zdrojem, kterého budeme využívat, bude ARP tabulka (resp. ARP cache). O vytvoření ARP tabulky se stará Address Resolution Protocol (ARP), který pracuje na pomezí druhé a třetí vrstvy

ISO/OSI modelu. Funguje na principu dotazu a odpovědi, kdy tazatel vytváří požadavek na hledanou IP adresu a připojuje k němu vlastní identifikaci. Tazateli odpovídá vlastník hledané IP adresy, který do odpovědi připojí své adresy na nižších vrstvách (typicky MAC adresa). Kvůli snížení počtu ARP dotazů si tazatel ukládá všechny výsledky do dočasné tabulky (ARP cache). Obsahem této tabulky je tedy seznam IP adres všech zjištěných sousedů s odpovídajícími fyzickými adresami (MAC). Data této tabulky nalezneme v IP-MIB, konkrétně tabulky ipNetToPhysicalTable a ipNetToMedia dostupné přes SNMP.

3 Návrh aplikace pro emulaci sítě

Svou aplikaci navrhuji pro použití s produktem Orion Network Performance Monitor firmy SolarWinds. Jednou z funkcí toho produktu je zjišťování síťové topologie vzájemně propojených zařízení v síti, její kalkulace a vizualizace na mapě. Využití mé aplikace bude pro automatizované testování funkce algoritmu pro výpočet topologie sítě. Topologické informace, které jsme si již popsali v teoretické části, získává Orion prostřednictvím protokolu SNMP.

S ohledem na tuto skutečnost zaměřuji emulaci síťových prvků na protokol SNMP a data týkající se topologie sítě. Pracovat budeme pouze s topologickými daty 2. vrstvy ISO/OSI modelu (linková vrstva), protože aktuální verze produktu Orion NPM zatím jiné nepodporuje. Pro implementaci mé aplikace navrhuji zvolit programovací jazyk C# a to z důvodu návaznosti na ostatní produkty firmy SolarWinds, které jsou rovněž v tomto jazyku napsány.

3.1 Funkční požadavky

V aplikaci budeme emulovat jednotlivá zařízení v síti. Každé zařízení má název, seznam svých rozhraní a informace o svých vlastnostech (podrobněji rozebráno v kapitole Komponenty aplikace). Jednotlivá rozhraní mají přiřazený název, fyzickou adresu a IP adresu. Potřebujeme také informace ohledně spojů, tj. toho jaké rozhraní kterého zařízení je ve spojení s některým dalším rozhraním jiného zařízení. Veškeré další funkční požadavky lze shrnout do následující tabulky.

Značení požadavku	*Popis*
PK01	*Aplikace umožní přidávání, mazání a editaci jednotlivých zařízení, jejich rozhraní, vlastností a vzájemných spojů mezi nimi*
PK02	*Aplikace bude schopna zobrazit aktuální vytvořenou síťovou topologii graficky na mapě*
PK03	*Aplikace bude schopna uložit stav nastavení do souboru – export*
PK04	*Aplikace bude schopna načíst stav nastavení ze souboru – import*
PK05	*Aplikace bude schopna spustit na základě nadefinované síťové topologie emulaci síťových prvků*
PK06	*Aplikace umožní porovnat, jestli výsledky zjištění topologie produktem Orion NPM odpovídají datům emulované topologie v naší aplikaci*

Tabulka 3.1 Hlavní funkční požadavky kladené na aplikaci pro emulaci sítě

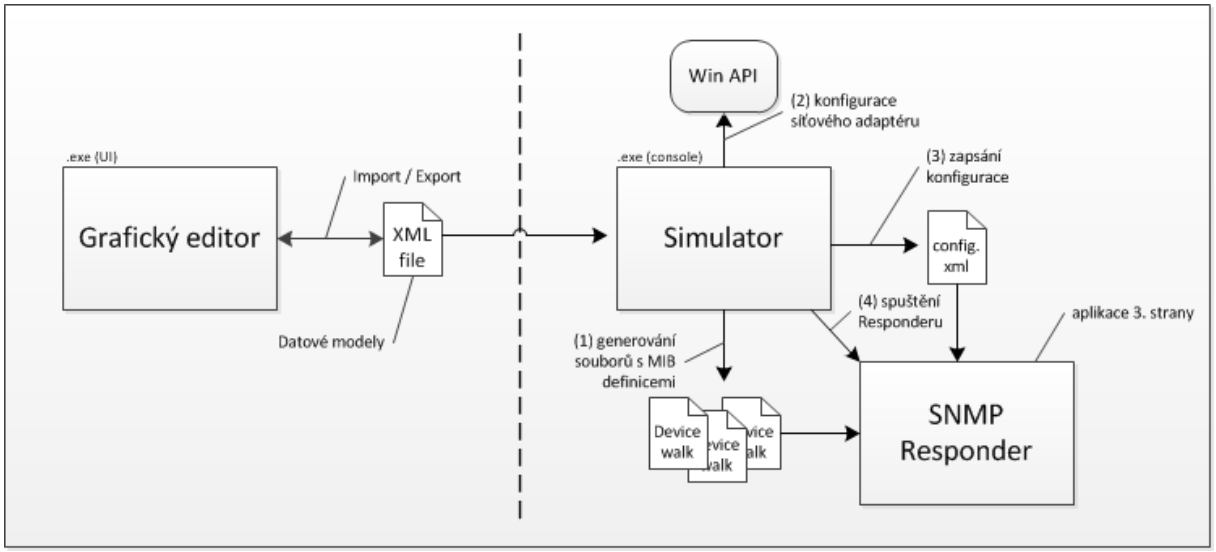
3.2 Nefunkční požadavky

Náš program je určen pro testovací účely interních pracovníků firmy SolarWinds, proto je spíše než na přívětivost uživatelského rozhraní kladen důraz na efektivitu a jednoduchost zadání potřebných dat pro rychlé spuštění emulace. Jelikož je aplikace určena pro potřeby zahraniční firmy, bude uživatelské rozhraní v anglickém jazyce.

Co se týče rozsahu emulované topologie, budeme preferovat použití spíše na „topologicky zajímavé“ scénáře menšího rozsahu (cca 10 až 20 propojených zařízení). Například problematické kombinace zapojení hlášené zákazníky, kde byl výsledek výpočtu topologie Orionem nepřesný či nesprávný. Aplikace není určena na emulování rozsáhlých sítí a na testování výkonu. Aplikace poběží v operačních systémech MS Windows a bude vyžadovat .NET Framework 4.0 nebo vyšší.

3.3 Komponenty aplikace

Návrh mé aplikace pro emulaci bude obsahovat 3 části – komponenty, které si v následujících odstavcích blíže rozebereme a popíšeme.



Obrázek 3.1: Diagram komponent aplikace

3.3.1 Grafický editor

Grafický editor bude obsahovat ovládací prvky, které umožní uživateli zadefinovat jednotlivá zařízení, jejich parametry a vzájemné spojení s ostatními prvky sítě zobrazené na mapě. Každé zařízení bude definováno názvem, vlastnostmi a seznamem svých rozhraní. Vlastnosti určují, jaký typ topologických dat bude zařízení poskytovat (CDP, LLDP, D-Bridge, Q-Bridge, ARP) a typ zařízení (směrovač, přepínač, pracovní stanice). Jednotlivá rozhraní budou obsahovat název, fyzickou adresu,

IP adresu a masku podsítě. Abychom věděli, co je s čím spojeno, potřebujeme mít seznam spojení na úrovni uzel - uzel a pod ním bližší informace na úrovni rozhraní – rozhraní. Tato data definující síť, by nám měla pro spuštění emulace postačit.

Uživatelské rozhraní grafického editoru umožní export nadefinovaného zapojení sítě do souboru typu XML. Uživatel bude mít možnost uložit si několik profilů nastavení sítě, které pak může importovat zpátky a dál libovolně měnit dle potřeby. Soubor XML bude obsahovat veškeré informace ohledně definic zařízení, rozhraní, vlastností a o způsobu propojení zařízení mezi sebou. S těmito daty bude dál pracovat 2. komponenta – Simulátor.

3.3.2 Simulátor

Simulátor bude mít na starost výpočet odpovídajících topologických dat nad daty ze souboru XML, které obsahují definici zařízení a stavu zapojení sítě. Výsledkem toho výpočtu budou topologické tabulky pro jednotlivá zařízení, ze kterých se vygenerují MIB definice (SNMP walks) pro každé zařízení. Dle počtu zařízení a jim odpovídajících IP adres provede Simulátor přes Win API konfiguraci síťového adaptéru. Jeden síťový adaptér je schopen odpovídat na více IP adresách, takže Simulátor tyto IP adresy přidá do konfigurace IP protokolu.

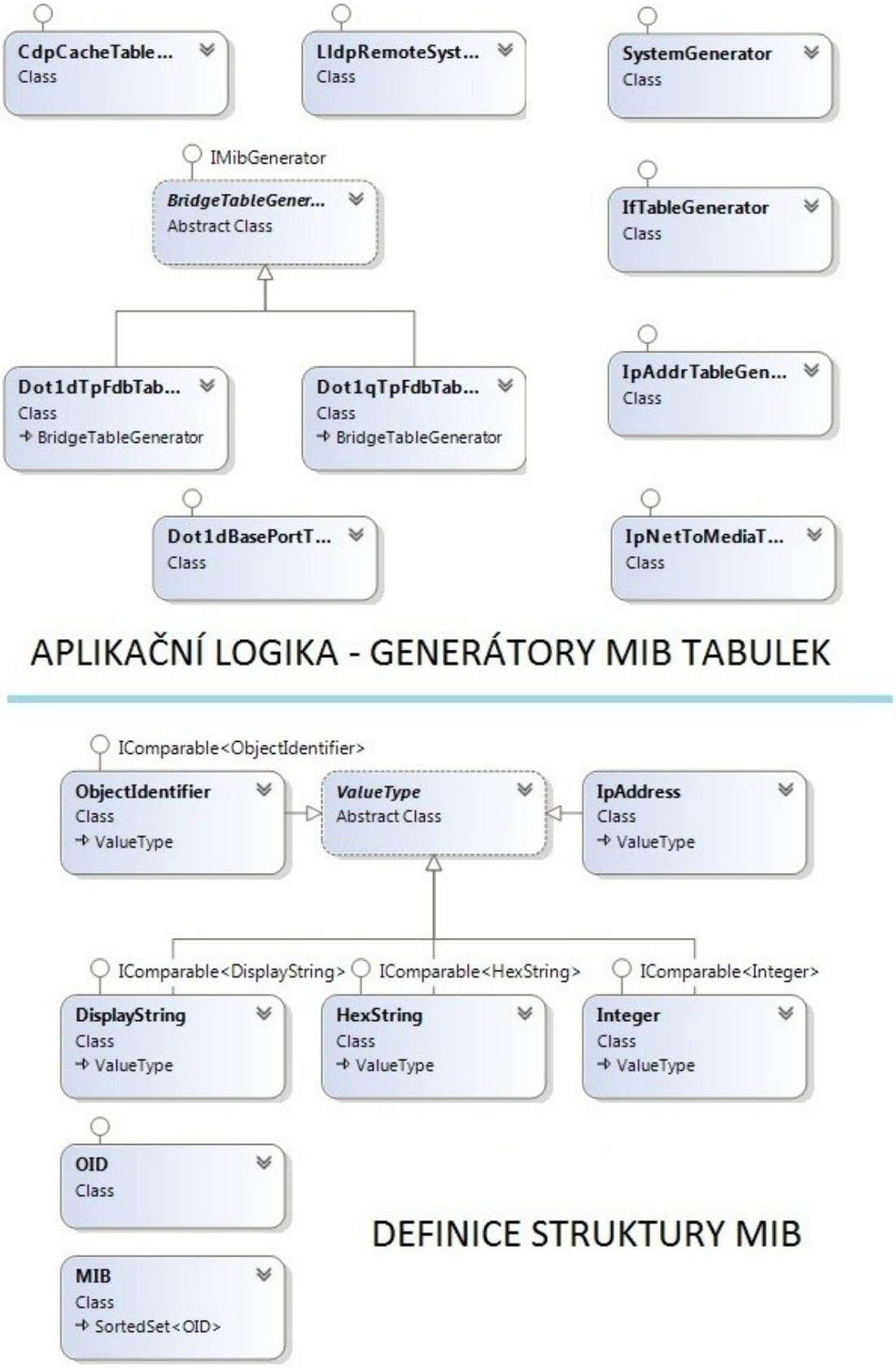
Na základě dvou předchozích akcí vytvoří Simulátor konfigurační soubor pro 3. komponentu – SNMP Responder. Následně spustí SNMP Responder. Funkcí Simulátoru je také porovnání výsledků vypočtené topologie z databáze Orionu s původními daty zadanými prostřednictvím Grafického editoru (XML soubor).

3.3.3 SNMP Responder

Jedná se o aplikaci 3. strany, která provádí emulaci zařízení. Princip fungování spočívá v tom, že na základně konfiguračního souboru, spustí Responder emulátory SNMP agentů na zadaných IP adresách, kteří odpovídají na SNMP dotazy. Hodnoty konkrétních OID pro odpovědi jsou načtena ze SNMP walk souborů, přiřazených k jednotlivým IP adresám. S touto aplikací budeme pracovat na binární úrovni, zdrojové kódy nelze zveřejnit.

3.4 Třídní diagramy

Obrázek 3.2: Třídní diagram pro GUI a datové modely



Obrázek 3.3: Třídní diagram pro aplikační logiku a strukturu MIB

4 Implementace

Tato kapitola popisuje způsob implementace celého programu. Přiblížíme si zde hlavní třídy, jejich funkci, popíšeme si použité algoritmy, využité MIB a ukážeme celou aplikaci. Pro vývoj jsem vybral platformu .NET Framework verze 4.0. Vývojové prostředí jsem zvolil Microsoft Visual Studio 2012. Aplikaci jsem pojmenoval TopologySimulator, což její zaměření vcelku vystihuje.

Pro implementaci aplikace jsem využil standardních součástí .NET frameworku. Pouze pro vizualizaci zapojení na mapě jsem využil knihovny Neuron. Neuron je komerční knihovna umožňující zobrazování digramů. Je v trial verzi zdarma s reklamou, nicméně pro naše potreby je naprosto dostačující.

4.1 Rozdělení aplikace a použité třídy

Celá aplikace je rozdělena do několika částí, které jsou funkčně odděleny. Základním stavebním kamenem je Grafické uživatelské rozhraní, pomocí kterého budeme ovládat celou aplikaci. Druhá část představuje reprezentaci veškerých dat (datové modely), se kterými budeme pracovat. Třetí částí je pak vlastní aplikační logika celého programu, která odpovídá funkcím komponenty Simulátor, popsané v kapitole návrhu. Nyní si pojďme blíže přiblížit třídy a jejich funkce pro jednotlivé části aplikace.

4.1.1 Třídy pro práci s GUI

O zobrazení grafického uživatelského rozhraní aplikace a jeho funkcionalitu se stará několik tříd. Třída MainForm tvoří hlavní okno celé aplikace, které je složeno z dílčích bloků s ovládacími prvky pro jednotlivé operace:

- DeviceDetails
- DeviceSelector
- InterfaceEditor
- InterfaceSelector
- LinkEditor
- LinkSelector
- MapView
- StatusBar

Tyto prvky oznamují změny třídě GuiState, která na základě těchto akcí publikuje vlastní definované události o stavu uživatelského rozhraní. Díky třídě GuiState se dozví ostatní části GUI, že

se něco odehrálo a mohou tak informace průběžně zapracovat. Druhé okno aplikace tvoří třída MibViewer, která slouží pro zobrazení všech OID z MIB vybraného zařízení, které budou pro toto zařízení aplikací vygenerovány. Výše uvedené třídy nejdeme ve jmenném prostoru TopologySimulator.Gui.

4.1.2 Třídy pro reprezentaci dat

Veškerá data, se kterými budeme pracovat, potřebujeme v aplikaci nějakým způsobem reprezentovat. K reprezentaci dat s vlastnostmi týkajících se jednotlivých zařízení, rozhraní a spojů, které budeme vytvářet, slouží třídy z jmenného prostoru TopologySimulator.Engine.Models. Jsou to tyto třídy:

- Capabilites
- Capability
- Connection
- Device
- Interface
- PhysicalLink

Třída Capabilites určuje, jaké vlastnosti (např. IP adresa, ARP cache, CAM tabulka, CDP data, LLDP data) mohou být dostupné pro jednotlivé typy zařízení (směrovač, přepínač, pracovní stanice). Krom výše uvedených tříd nemohu opomenout také další důležité třídy:

- Responder
- Simulation
- SimulationRunInfo

Třída Simulation je zodpovědná za vytváření kolekcí jednotlivých zařízení, rozhraní, spojů a zabezpečuje integritu datového modelu simulace. Každý nový objekt je zaindexován a vložen do kolekce podle unikátního ID. Pod tímto ID je pak přístupný. Třída také hlídá jednotlivé datové objekty a při změně dává pomocí událostí vědět ostatním, že nastala změna. Prvky grafického rozhraní na základě těchto událostí změnu zapracují (např. změní popis zařízení na mapě).

Třída Responder reprezentuje data pro komponentu Responder. Jedná se o IP adresu a cestu k souboru s kompletní MIB definicí (SNMP walk). Pro každé zařízení je vytvořena jedna instance třídy Responder. Poslední zmíněná třída SimulationRunInfo obsahuje kolekci objektů třídy Responder. Na základě této struktury je pak vytvořen konfigurační soubor (config.xml), pomocí kterého se pak spustí program SNMP Responder (komponenta 3. strany) pro emulaci všech zařízení.

Abychom mohli generovat MIB tabulky, kterými budeme plnit SNMP definice jednotlivých zařízení (SNMP walks), vytvořili jsme strukturu určující jak má formát dat v těchto souborech vypadat. Slouží k tomu následující třídy:

- DisplayString
- HexString
- Integer
- IpAddress
- MIB
- ObjectIdentifier
- OID
- ValueType

Třída ValueType je bázová třída pro všechny datové typy definované v SNMP. Třída OID reprezentuje dvojici ObjectIdentifier (např. 1.3.6.1) a ValueType (např. INTEGER = 42) a tím definuje jeden řádek v definici MIB zařízení (SNMP walk). Třída MIB se stará o to, aby se jednotlivá OID neopakovala a byla řazena vzestupně dle obvyklé konvence struktury definic. Ostatní třídy slouží pro reprezentaci a unifikaci formátu datového typu hodnot, které mohou jednotlivá OID nabývat. Všechny tyto třídy jsou uloženy v jmenném prostoru TopologySimulator.Engine.Snmp.

4.1.3 Třídy pro aplikační logiku

Aplikační logika je tvořena třídami generující jednotlivé MIB tabulky s odpovídajícími OID, ze kterých je postupně složena kompletní MIB definice pro zařízení. Za pozornost zde stojí algoritmy pro generování jednotlivých topologických dat, které si podrobně popíšeme v následující kapitole. V jmenném prostoru ToplogySimulator.Engine.Generators najdeme tyto třídy:

- BridgeTableGenerator
- CdpCacheTableGenerator
- Dot1dBasePortTableGenerator
- Dot1dTpFdbTableGenerator
- Dot1qTpFdbTableGenerator
- IfTableGenerator
- IMibGenerator
- IpAddrTableGenerator
- IpNetToMediaTableGenerator
- LldpRemoteSystemsDataGenerator

- SystemGenerator

Třídy SystemGenerator, ifTableGenerator a IpaddrTableGenerator generují MIB tabulky týkající se se zařízení jako takového (jsou zde informace např. název a popis zařízení, seznam rozhraní apod.). Ostatní zmíněné třídy mají na starost generování tabulek s topologickými daty pro jednotlivá zařízení. Vygenerování kompletní MIB struktury definující zařízení (SNMP walk) a její uložení do souboru na disk se stará třída SimulationEngine. O které tabulky a přesná OID jde, můžeme najít v třídě OIDs, nicméně co přesně jednotlivá OID nesou za informaci si probereme zvlášť v kapitole 4.3 – Využité informace z MIB databáze.

Poslední částí týkající se aplikační logiky jsou třídy pro porovnání a vyhodnocení emulované topologie z naší aplikace se zjištěnou topologií produktem Orion Network Performance Monitor. Jedná se o následující třídy:

- CompareResult
- OrionDataLoader
- OrionTopologyConnection

Třída OrionTopologyConnection reprezentuje zjištěná topologická data uložená v Orionu. Třída OrionDataLoader načte odpovídající topologická data z Orionu a porovná je s daty nadefinovanými prostřednictvím naší aplikace. Třída CompareResult obsahuje 2 kolekce - chybějící spoje a přebývající spoje.

4.2 Použité algoritmy

Co se týče použitých algoritmů, pozornost budu věnovat části generování dat jednotlivých MIB tabulek. Pro vytvoření statických informací ohledně uzlů, jejich rozhraní a vlastností postačí triviální algoritmus, který zadefinovaná data vezme a vytvoří k nim odpovídající řádky OID. Pro rozhraní se vytvoří indexy, pomocí kterých se propojí s ostatními informacemi z jiných tabulek, dle obvyklých definic MIB. Jedná se o tabulky SNMPv2-MIB, IF-MIB a IP-MIB.

Hlavní zajímavou algoritmickou částí, kterou si podrobně projdeme, je způsob procházení celé struktury sítě, hledání sousedících uzlů a ukládání informací o nich. V návaznosti se potom dle těchto informací generují odpovídající topologické tabulky s daty (BRIDGE-MIB, CISCO-CDP-MIB, LLDP-MIB).

4.2.1 Algoritmus prohledání sítě

Jelikož struktura sítě odpovídá grafu, využijeme k prohledání sítě známého grafového algoritmu BFS (Breadth-first search) – Prohledávání do šířky. Na začátku je třeba určit uzel (tzv. seedDevice), ze kterého budeme procházet další okolní sousedy a ke kterému se aktuální výpočet cest

k ostatním uzlům v rámci podsítě vztahuje. Prohledáváme do šířky, takže v prvním kroku zjistíme bezprostřední sousedy, které si uložíme do seznamu, co by následující uzly v pořadí pro další úroveň prohledávání (currentHopCandidates). Tento seznam se pak následně projde a pro každý uzel se opět provede prohledání okolních sousedů o další úroveň a vytvoří se seznam (nextHopCandidates) dalších uzlů v okolí, které lze dále projít. Po projití posledního uzlu z currentHopCandidates se nextHopCandidates prohlásí za currentHopCandidates a pokračuje se iterativně dál. Algoritmus končí v momentě, kdy prošel všechny uzly ze seznamu currentHopCandidates a seznam nextHopCandidates je prázdný. Nezapomeňme podotknout, že si celou dobu evidujeme seznam navštívených uzlů, který se průběžně aktualizuje. Je to z toho důvodu, abychom předešli smyčkám a zbytečně dokola neprocházeli stejné uzly.

Aby byl algoritmus použitelný obecně a mohli jsme ho využít pro generování všech typů topologických dat, zvolil jsem pro algoritmus několik parametrů. Prvním parametrem je funkce (deviceSelector), která dle definovaných podmínek určuje, zdali se výpočet na procházeném uzlu zastaví či nikoliv a jestli se uzel přidá do seznamu. Je to tedy dvojice uzel a odpovídající stav, který může nabývat 3 hodnot – zastav (stop), přidej-zastav (IncludeStop), přidej-pokračuj (IncludeContinue). Tyto podmínky se dle typu topologických dat liší. Druhým parametrem je číslo, určující jak hluboko do šířky budeme prohledávat. Tj. Délka prohledávané cesty z výchozího uzlu (maxHopCount). Poslední parametr je již zmíněné výchozí zařízení (seedDevice), ze kterého začneme procházet.

Výstup algoritmu tvoří seznam všech dosažitelných uzlů od výchozího zařízení a cesty k nim (tj. po jakých spojích jsou uzly dostupné – najdeme zde informace o jednotlivých propojených rozhraních). Tento algoritmus se v rámci vygenerování kompletních MIB definicí spustí pro každé nadefinované zařízení zvlášť. Zdrojový kód celého algoritmu je uveden níže (Obrázek 4.1).

```
public IEnumerable<PathInfo> GetDeviceNeighbours(Device seedDevice, int hops, Func<Device, SelectorOutcome> deviceSelector)
{
    int remainingHops = hops;
    var results = new Dictionary<int, PathInfo>();
    var visitedDevices = new HashSet<int>();
    var currentHopCandidates = new List<Device>();

    // register seed as visited device (avoid loops to seed)
    visitedDevices.Add(seedDevice.ID);

    // add seed as next hop candidate to begin
    currentHopCandidates.Add(seedDevice);

    while (remainingHops > 0 && currentHopCandidates.Any())
    {
        var nextHopCandidates = new List<Device>();

        // execute one iteration of Breadth-first search
        foreach (var candidate in currentHopCandidates)
        {
            foreach (var neighbourPathInfo in GetDirectDeviceNeighbours(candidate))
            {
                // register device as visited (avoid multiple passes)
                if (!visitedDevices.Add(neighbourPathInfo.TargetDeviceID))
                {
                    continue; // device was processed already
                }

                Device otherDevice;
                if (this.TryGetDevice(neighbourPathInfo.TargetDeviceID, out otherDevice))
                {
                    var selRes = deviceSelector(otherDevice);

                    if (selRes == SelectorOutcome.Stop)
                    {
                        continue; // other device can't be processed, because it was denied by provided selector function
                    }

                    // save path to other device into result
                    PathInfo path;

                    PathInfo pathToCurrent;
                    if (results.TryGetValue(candidate.ID, out pathToCurrent))
                    {
                        path = PathInfo.Append(pathToCurrent, otherDevice.ID, neighbourPathInfo.Path.First());
                    }
                    else
                    {
                        path = new PathInfo(otherDevice.ID, neighbourPathInfo.Path);
                    }
                    results.Add(otherDevice.ID, path);

                    if (selRes == SelectorOutcome.IncludeContinue)
                    {
                        // register other devices to next hop candidate
                        nextHopCandidates.Add(otherDevice);
                    }
                }
            }

            currentHopCandidates = nextHopCandidates;
            remainingHops--;
        }
    }

    return results.Values;
}
```

Obrázek 4.1: Zdrojový kód algoritmu pro prohledání sítě

4.2.2 Algoritmus pro generování ipNetToMedia tabulky

Pro vygenerování ipNetToMedia tabulky potřebujeme získat seznam MAC adres z podsítě, ve které se daný uzel nachází. Tyto MAC adresy se pak dále spárují s odpovídajícími IP adresami uzlů.

Ze znalosti principů počítačových sítí víme, že jednotlivé podsítě ohraničují zařízení typu směrovač. Do podmínky pro procházení sítě (Obrázek 4.2) tedy nadefinujeme, aby se výpočet u směrovačů zastavil.

```
private static readonly Func<Device, Simulation.SelectorOutcome> SelectorFunc = (d) =>
{
    if (d.Role == Device.Roles.Router)
    {
        return Simulation.SelectorOutcome.IncludeStop;
    }
    else
    {
        return Simulation.SelectorOutcome.IncludeContinue;
    }
};
```

Obrázek 4.2: Zdrojový kód podmínky pro zastavení výpočtu u směrovače

Následně využijeme algoritmu procházení sítě, který zavoláme s touto podmínkou. Jako druhý parametr pro určení hloubky vyhledávání volím hodnotu Int32.MaxValue. Což je dostačující hodnota pro případné prohledání větší sítě. Pro každý uzel ze získaného seznamu všech dosažitelných uzlů s cestami kontrolujeme, zdali jednotlivé uzly ARP podporují. Pokud ne, vrátíme pro daný uzel prázdnou MIB tabulku. Dále si postupně ukládáme informace vždy o prvním spoji ze zdrojového uzlu a informace o posledním spoji vedoucím do cílového uzlu. Tohle nám zaručí, že při postupném procházení cest sítě bude každý spoj zpracován jen jednou. Jelikož se informace tvoří postupně přes cyklus foreach, získáme tímto způsobem veškerá potřebná data, která se odpovídajícím způsobem vkládají do jednotlivých OID.

Při generování dat nám vzniká ještě jeden problém a to takový, že jednotlivé spoje máme orientované. Rozhraní u spojů jsou definována jako zdrojové a cílové. Při procházení sítě, pak dle směru mohou být tyto rozhraní u linek prohozená opačně. Proto si tuto skutečnost u jednotlivých spojů zaznamenáváme a u generování dat pak pro opačně orientované linky rozhraní prohodíme tak, aby data s pohledu procházeného uzlu korespondovaly správně.

4.2.3 Algoritmus pro generování CAM tabulky

CAM tabulka obsahuje párování fyzických adres s rozhraním (resp. portem), přes který je možné na tyto fyzické adresy posílat rámce. Tj. potřebujeme získat seznam MAC adres z podsítě a ty pak následně přiřadit k lokálním portům tak, aby bylo možné se přes odpovídající porty na tyto MAC adresy dostat. Předpokládám, že se můžeme dostat všude. Pokročilejší logiku rozdělení linek pomocí STP zde neřeším. Algoritmus prohledávání sítě zavoláme stejně jako v případě ipNetToMedia tabulky. Budeme tedy prohledávat stejné okolí s tím rozdílem, že budeme ukládat jiná data. U zdrojového uzlu

budeme kontrolovat zdali CAM podporuje či nikoliv. Pokud CAM není podporováno, vrátíme pro daný uzel prázdnou MIB tabulku. Výpočet se zastaví rovněž u zařízení typu směrovač.

4.2.4 Algoritmus pro generování CDP a LLDP topologických dat

V případě CDP a LLDP topologických dat potřebujeme získat pro každé zařízení informace o přímo připojených uzlech v okolí. U zařízení budeme kontrolovat jestli podporuje CDP či LLDP. A to jak u zdrojového uzlu tak u okolních. V případě, že CDP či LLDP nepodporuje zdrojový uzel, zůstane odpovídající tabulka pro uzel prázdná. Pokud CDP či LLDP nepodporuje některý z procházených uzlů z okolí, výpočet se u něj zastaví a nebudeme jej brát v potaz. Algoritmus procházení sítě tedy zavoláme s touto podmínkou. Co se týče parametru pro hloubku procházení, zvolíme hodnotu 1. Ta nám říká, že budeme procházet uzly pouze do vzdálenosti 1, což jsou přímo připojené uzly v nejbližším okolí. Algoritmus jinak funguje obdobně jako u předchozích s tím, že data jsou generována do odpovídajících tabulek pro CDP a LLDP topologická data. Konkrétně jsou to tabulky cdpCacheTable a lldpRemoteSystemData.

4.3 Využité informace z MIB databáze

V této kapitole si uvedeme MIB tabulky, které jsem použil ve své aplikaci. Aby bylo zřejmé z jakých informací, poskytovaných protokolem SNMP, generujeme kompletní obsah MIB definic pro zařízení, zaměřím se také na popis obsahu jednotlivých OID. Využitá OID jsem zvolil odpovídajícím způsobem dle podpory produktem Orion Network Performance Monitor.

4.3.1 SNMPv2-MIB

Jedná se o hlavní systémovou tabulku s kořenovým OID 1.3.6.1.2.1.1, která obsahuje převážně statické informace o uzlu samotném (např. název, popis, typ, umístnění atd.). Z této tabulky nám postačí použít informace ohledně názvu uzlu a informace ohledně typu zařízení. Žádné další údaje Orion pro výpočet topologie nevyžaduje.

Název objektu	OID	popis
sysObjectID	1.3.6.1.2.1.1.2.0	typ zařízení/subsystému uvedený výrobcem
sysName	1.3.6.1.2.1.1.5.0	administrativně přidělený název spravovaného uzlu

Tabulka 4.1: Využité OID z SNMPv2-MIB

Obsahem SysObjectID je hodnota z tabulky SNMPv2-SMI s kořenovým OID 1.3.6.1.4.1, resp. hodnota OID některého z jejích podstromů. Například výrobce Microsoft má přiřazený podstrom 1.3.6.1.4.1.311. Výsledná hodnota OID v sysObjectID by tak mohla být například 1.3.6.1.4.1.311.1.1.3.1.1, což nám říká, že se jedná o WindowsNT pracovní stanici.

4.3.2 **IF-MIB**

Obsahem IF-MIB jsou tabulky s informacemi ohledně síťových rozhraních dotazovaného uzlu. Kořenové OID tabuky IF-MIB je 1.3.6.1.2.1.2. Najdeme zde tabulku ifNumber (1.3.6.1.2.1.2.1.0), která udává celkový počet rozhraní, kterým dotazovný uzel disponuje. Využití této tabulky pro naše účely však není nutné. Další tabulky, které již využívat budeme, jsou tabulka ifTable a novější ifXTable. Tyto tabulky evidují informace ohledně jednotlivých rozhraní.

Název objektu	OID	Popis
ifIndex	1.3.6.1.2.1.2.2.1.1	číselný identifikátor pro jednotlivá rozhraní
ifDescr	1.3.6.1.2.1.2.2.1.2	text s informací o rozhraní (název, výrobce)
ifType	1.3.6.1.2.1.2.2.1.3	typ rozhraní
ifPhysAddress	1.3.6.1.2.1.2.2.1.6	fyzická (MAC) adresa rozhraní
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	konfigurovaný stav rozhraní
ifOperStatus	1.3.6.1.2.1.2.2.1.8	aktuální provozní stav rozhraní
ifName	1.3.6.1.2.1.31.1.1.1.1	název rozhraní
ifAlias	1.3.6.1.2.1.31.1.1.1.18	uživatelem definovatelný název rozhraní

Tabulka 4.2: Využité OID z IF-MIB

Jako identifikátor v rámci tabulky se používá ifIndex. Tohoto identifikátoru se hojně využívá také v dalších MIB, kde ho lze dobře využít pro spojování dat z ruzných tabulek, slouží jako primární klíč do tabulky ifTable. Hodnota objektu ifType, udávající typ rozhraní (např. FastEthernet, GigabitEthernet, Loopback Interface, Tunnel atd.), je definována organizací Internet Assigned Numbers Authority (IANA), což jsou čísla v rozmezí 1-234. U rozhraní v naší aplikaci používám výchozí hodnotu 117, která odpovídá rozhraní typu GigabitEthernet. Objekt ifAdminStatus udává konfigurovaný stav rozhraní, tj. stav, který je požadován resp. očekáván. Tento stav může být explicitně změněn například administrátorem. Zato objekt ifOperStatus udává faktický stav rozhraní. Stavy obou hodnot by si měly odpovídat, nicméně mohou se lišit. Například pokud administrátor rozhraní povolí v konfiguraci, ale nezapojí síťový kabel, bude ifAdminStatus roven hodnotě 1 (Up) a

ifOperStatus bude roven hodnotě 2 (Down). Spojení na fyzické vrstvě není navázáno, ale rozhraní je v konfiguraci povoleno. Naše aplikace bude vždy nastavovat oba stavy na hodnotu 1.

4.3.3 IP-MIB

IP-MIB je dostupná přes kořenové OID 1.3.6.1.2.1.4 a jsou v ní uloženy informace ohledně konfigurace IP protokolu. Najdeme zde například síťové adresy přiřazené jednotlivým rozhraním, masku podsítě, směrovací informace, data z protokolu ARP a mnoho dalších. A to jak pro protokol verze IPv4, tak pro verzi IPv6. Já se rozhodl pro využití dat pouze pro IPv4.

Z IP-MIB využívám 2 tabulky, ipAddrTable dostupnou přes OID 1.3.6.1.2.1.4.20 a ipNetToMediaTable dostupnou přes OID 1.3.6.1.2.1.4.22.

Název objektu	OID	Popis
ipAdEntIfIndex	1.3.6.1.2.1.4.20.1.2	index odpovídajícího rozhraní pro IP adresu v záznamu
ipNetToMediaIfIndex	1.3.6.1.2.1.4.22.1.1	index odpovídajícího rozhraní
ipNetToMediaPhysAddress	1.3.6.1.2.1.4.22.1.2	fyzická (MAC) adresa rozhraní
ipNetToMediaNetAddress	1.3.6.1.2.1.4.22.1.3	IP adresa korespondující s fyzickou adresou
ipNetToMediaType	1.3.6.1.2.1.4.22.1.4	typ mapování

Tabulka 4.3: Využité OID z IP-MIB

Jak již název napovídá, tabulka ipAddrTable obsahuje informace ohledně IP adres přiřazených na jednotlivých rozhraních. Z této tabulky použijeme objekt ipAdEntIfIndex, který jako součást svého OID obsahuje přiřazenou IP adresu pro rozhraní (např. 1.3.6.1.2.1.4.20.1.2.10.10.10.1). Hodnota objektu pak představuje číslo ifIndexu (definován v IF-MIB), odpovídajícího rozhraní ke kterému IP adresa patří.

Druhou využívanou tabulkou je tabulka ipNetToMediaTable. Ta obsahuje párování síťových adres s fyzickými. Tyto data se do tabulky dostávají z výsledků provedených ARP dotazů. IP adresa je u všech objektů rovněž součástí OID podobně jako v případě ipAdEntIfIndex. Takže je nutné ji rozpársovat. Objekt ipNetToMediaType určuje, zda-li se jedná o statický, dynamický nebo neplatný záznam. V našem případě budeme používat statické záznamy, kterým odpovídá hodnota 4.

Ve výsledku tedy z IP-MIB vygeneruji seznam všech adres na rozhraních pro uzel a seznam jeho sousedů na třetí vrstvě ISO/OSI modelu.

4.3.4 BRIDGE-MIB, Q-BRIDGE-MIB

Obsahuje tabulky z prostředí síťových přepínačů a daty z STP (Spanning Tree Protocol), kořenové OID celé báze je 1.3.6.1.2.1.17. První využitou tabulkou je dot1dBasePortTable, která je dostupná přes OID 1.3.6.1.2.1.17.1.4. Obsahuje mapování portů na rozhraní (číslo portu - ifIndex). Druhou využitou tabulkou je dot1dTpFdbTable dostupnou přes OID 1.3.6.1.2.1.17.4.3. Tato tabulka zpřístupňuje záznamy z CAM tabulky, která obsahuje párování fyzických adres s rozhraním (resp. portem), na kterém je uzel s vybranou MAC adresou připojen (viz. Kapitola 2.4.3). Třetí využitou tabulkou je dot1qTpFdbTable (1.3.6.1.2.1.17.7), která je obdobou dot1dTpFdbTable.

Název objektu	OID	popis
dot1dBasePort	1.3.6.1.2.1.17.1.4.1.1	číslo portu
dot1dBasePortIfIndex	1.3.6.1.2.1.17.1.4.1.2	index odpovídajícího rozhraní
dot1dTpFdbAddress	1.3.6.1.2.1.17.4.3.1.1	MAC adresa zařízení připojeného na port
dot1dTpFdbPort	1.3.6.1.2.1.17.4.3.1.2	číslo portu, na kterém je MAC adresa připojena
dot1dTpFdbStatus	1.3.6.1.2.1.17.4.3.1.3	stav záznamu
dot1qTpFdbAddress	1.3.6.1.2.1.17.7.1.2.2.1.1	MAC adresa zařízení připojeného na port
dot1qTpFdbPort	1.3.6.1.2.1.17.7.1.2.2.1.2	číslo portu, na kterém je MAC adresa připojena
dot1qTpFdbStatus	1.3.6.1.2.1.17.7.1.2.2.1.3	stav záznamu

Tabulka 4.4: Využité OID z BRIDGE-MIB a Q-BRIDGE-MIB

dot1dTpFdbStatus a dot1qTpFdbStatus udává stav záznamu (číslo 1-5). Naše aplikace nastaví tyto stavy vždy na hodnotu 3, což znamená „naučený“ (learned). To znamená, že záznam byl pořízen pomocí analýzy komunikace s okolními zařízeními. Fyzická adresa je také decimálně zakódována jako součást OID u všech využitých objektů z tabulek dot1dTpFdbTable a dot1qTpFdbTable. Slouží jako index řádku.

4.3.5 CISCO-CDP-MIB

CISCO-CDP-MIB je adresovatelná přes OID 1.3.6.1.4.1.9.9.23. Obsahuje data získaná z CDP protokolu určeného pro zařízení Cisco (viz. Kapitola 2.4.1). Využil jsem tabulku cdpCacheTable (1.3.6.1.4.1.9.9.23.1.2.1), která obsahuje informace o přímo připojených sousedech.

Název objektu	OID	popis
cdpCacheAddressType	1.3.6.1.4.1.9.9.23.1.2.1.1.3	označení typu adresy
cdpCacheAddress	1.3.6.1.4.1.9.9.23.1.2.1.1.4	vlastní adresa (většinou IP)
cdpCacheDeviceId	1.3.6.1.4.1.9.9.23.1.2.1.1.6	identifikátor sousedního uzlu
cdpCacheDevicePort	1.3.6.1.4.1.9.9.23.1.2.1.1.7	název vzdáleného rozhraní připojeného sousedícího uzlu

Tabulka 4.5: Využité OID z CISCO-CDP-MIB

V případě, že hodnota v poli cdpCacheAddressType je rovna 1, pak je v cdpCacheAddress uložena IP adresa (hexadecimálně zakódována). Nicméně reálně se zde mohou objevovat i jiné typy adres (čísla 1-25). Ty jsou ale pro náš případ nevyužitelné, takže budu v poli cdpCacheAddressType pracovat s hodnotu 1. V poli cdpCacheDeviceId budu ukládat název sousedícího uzlu (sysName), protože Orion jiné hodnoty nepodporuje. Reálně se zde však může objevovat také například MAC adresa, seroivé číslo zařízení nebo jiné označení zařízení.

4.3.6 **LLDP-MIB**

Poslední využitou MIB je LLDP-MIB, která je dostupná přes kořenové OID 1.0.8802.1.1.2. Obsahuje tabulky s informacemi získané prostřednictvím LLDP protokolu (viz. Kapitola 2.4.2). Využít jsem se rozhodl tabulky LldpRemManAddrTable (1.0.8802.1.1.2.1.4.2) a LldpRemTable (1.0.8802.1.1.2.1.4.1). Tyto tabulky obsahují informace o přímo připojených sousedech.

Název objektu	OID	popis
lldpRemManAddrIfSubtype	1.0.8802.1.1.2.1.4.2.1.3	způsob identifikace vzdáleného rozhraní
lldpRemManAddrIfId	1.0.8802.1.1.2.1.4.2.1.4	hodnota identifikující vzdálené rozhraní
lldpRemChassisIdSubtype	1.0.8802.1.1.2.1.4.1.1.4	způsob identifikace vzdáleného uzlu
lldpRemChassisId	1.0.8802.1.1.2.1.4.1.1.5	identifikace vzdáleného uzlu
lldpRemPortId	1.0.8802.1.1.2.1.4.1.1.7	portId/ifName vzdáleného rozhraní sousedícího uzlu
lldpRemPortDesc	1.0.8802.1.1.2.1.4.1.1.8	název vzdáleného rozhraní připojeného sousedícího uzlu
lldpRemSysName	1.0.8802.1.1.2.1.4.1.1.9	název sousedního uzlu

Tabulka 4.6: využité OID z LLDP-MIB

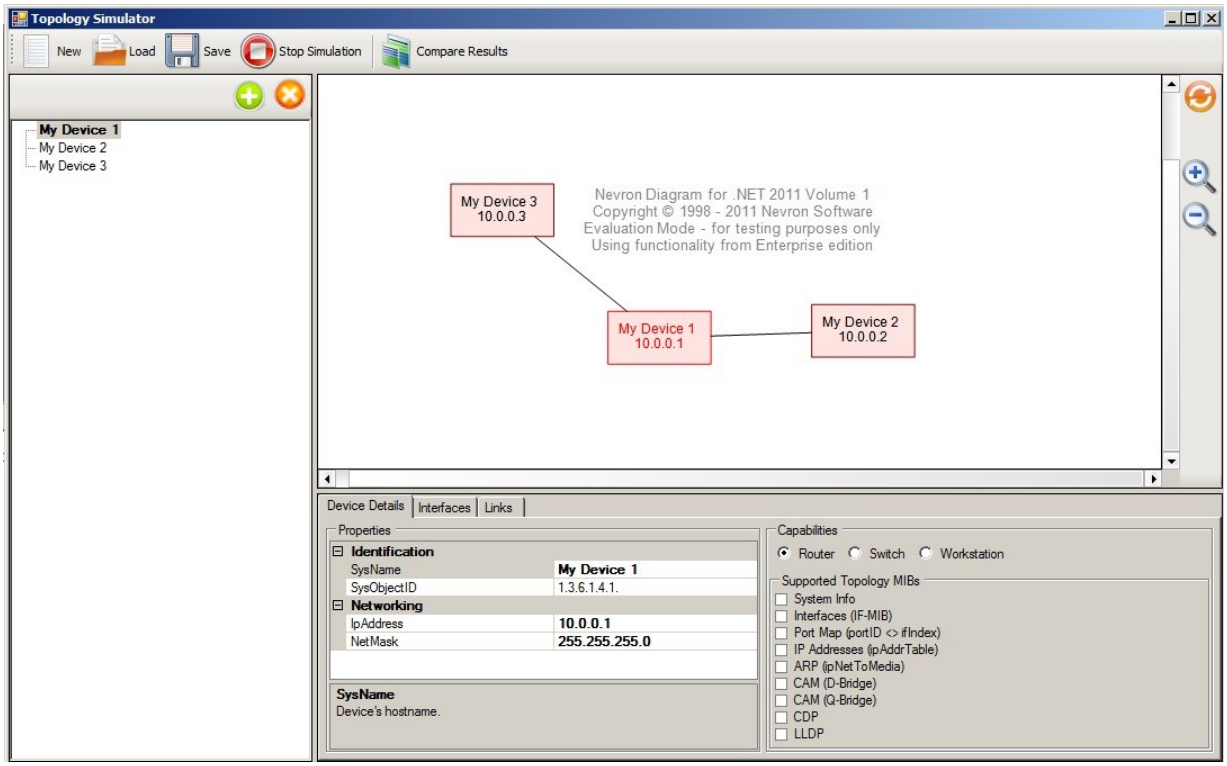
Objekt lldpRemManAddrIfSubtype udává, jakým typem informace budeme rozpoznávat vzdálené rozhraní sousedního uzlu. Zvolili jsme hodnotu 2, která říká, že rozhraní bude identifikováno pomocí ifIndexu. Objekt lldpRemManAddrIfId v našem případě udává číslo ifIndexu vzdáleného rozhraní, určeného na správu zařízení. Za tento ifIndex jsme zvolili hodnotu 0, což je běžná hodnota. Obdobou lldpRemManAddrIfSubtype je lldpRemChassisIdSubtype, který udává jakým typem informace budeme rozpoznávat vzdálený uzel. Zde jsme zvolili hodnotu 5, která říká, že uzel bude identifikován přes síťovou adresu. Síťová adresa vzdáleného uzlu je v hexadecimálním tvaru obsažena v objektu lldpRemChassisId. Objekt lldpRemPortId složí k identifikaci vzdáleného rozhraní, může zde být hodnota ifName (název rozhraní) nebo portId (číslo portu) v případě, že není dostupný ifName. Hodnoty jsou v textové podobě.

Objekty lldpRemManAddrIfSubtype a lldpRemManAddrIfId mají zajímavou strukturu jednotlivých OID. Jeho součástí jsou totiž další zakódované informace. Jedná se časovou značku reprezentující aktuální čas, interpolovanou do kladného intervalu v rozsahu datového typu Integer. Dále identifikace portu lokálního rozhraní, remIndex, což je revize záznamu v rámci LLDP (základní index 1.1). Poté určení typu následujícího identifikátoru zařízení, zde bude použijeme hodnotu 4, což odpovídá IP adrese. A nakonec následuje IP adresa samotná. Výsledné OID objektu lldpRemManAddrIfSubtype pak bude vypadat například takto 1.0.8802.1.1.2.1.4.2.1.3.1275972455.4.1.1.4.10.0.0.2

4.4 Aplikace a její funkce

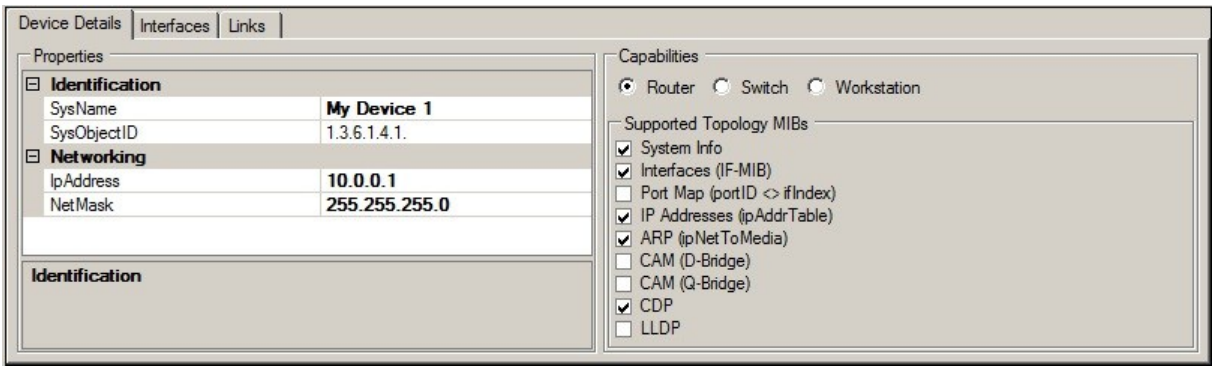
V této kapitole si ukážeme výslednou aplikaci a popíšeme její funkcionalitu. Po spuštění aplikace se zobrazí hlavní okno celé aplikace, které je rozděleno do 3 oddílů s ovládacími prvky (Obrázek 4.3). Zde si definujeme zařízení s rozhraními, jejich propojení, parametry a zadáváme tak celou síťovou topologii, která bude následně emulována.

V levé části okna je oddíl se seznamem všech zařízení (uzlů sítě). Tento seznam je po spuštění aplikace prázdný. Pro přidání nového zařízení slouží zelené tlačítko se znakem plus. Po stisknutí tohoto tlačítka se nové zařízení krom přidání do seznamu také graficky znázorní na mapce, která tvoří největší oddíl. Oranžovým tlačítkem X zařízení odebereme.



Obrázek 4.3: Hlavní okno aplikace Topology Simulator

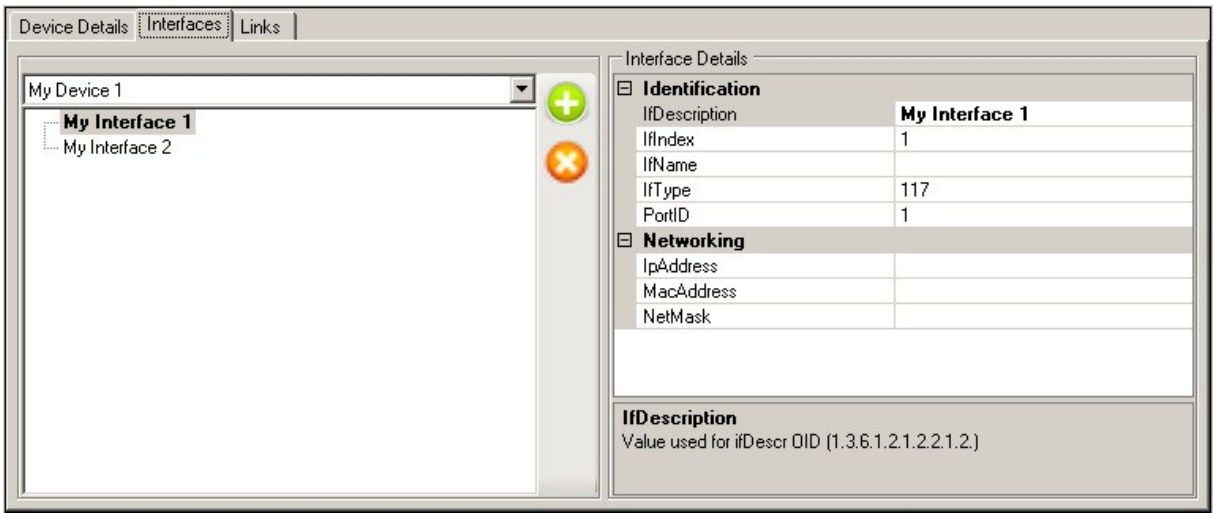
Ve spodní části je oddíl s informacemi a detaily ohledně nastavení parametrů. Obsahuje 3 karty (záložky) podle toho, čeho se informace týkají. První karta – Device Details obsahuje podrobnosti ohledně zařízení (Obrázek 4.4). Zde lze nastavit název zařízení (SysName), typ zařízení (SysObjectID - obsahuje informaci o výrobci a běžícím systému), IP adresu a masku podsítě. Uvedená IP adresa znamená adresu, pod kterou bude možné zařízení přidat do Orion Network Performance Monitor. Dále můžeme zvolit, zda si jedná o zařízení typu směrovač, přepínač nebo pracovní stanici (Capabilities). V neposlední řadě je třeba zvolit, které MIB tabulky bude zařízení generovat. Na tuto volbu je dobré nezapomenout, protože pokud není nic vybráno, nebudou k dispozici žádná data pro generování obsahu MIB definic.



Obrázek 4.4: Oddíl s informacemi o zařízení – karta Device Details

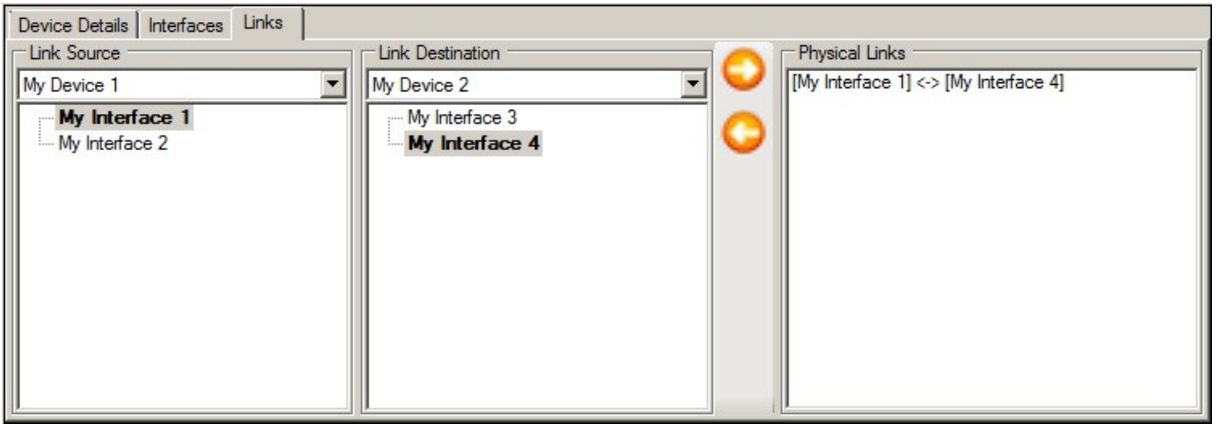
Co se týče volby generovaných MIB tabulek, tak rozdělení odpovídá struktuře, popsané v předchozí kapitole (Využité informace z MIB databáze).

Druhá karta – Interfaces zobrazuje informace a nastavení ohledně rozhraní (Obrázek 4.5). V levé části nalezneme seznam všech rozhraní pro zvolený uzel. Podobně jako tomu bylo u uzlů, je seznam po vytvoření prázdný. Naplnit jej můžeme zeleným tlačítkem se znakem plus. Tlačítko X rozhraní ze seznamu odstraní. V pravé části jsou zobrazené podrobnosti o jednotlivých rozhraních. Zde jsme schopni nastavit název rozhraní (IfDescription, popřípadě IfName), typ portu (ifType), číslo portu (PortID), IP adresu rozhraní, fyzickou adresu a síťovou masku. Ve spodní části jsou zobrazeny podrobnější vysvětlivky k jednotlivým údajům.



Obrázek 4.5: Oddíl s informacemi o rozhraní – karta Interfaces

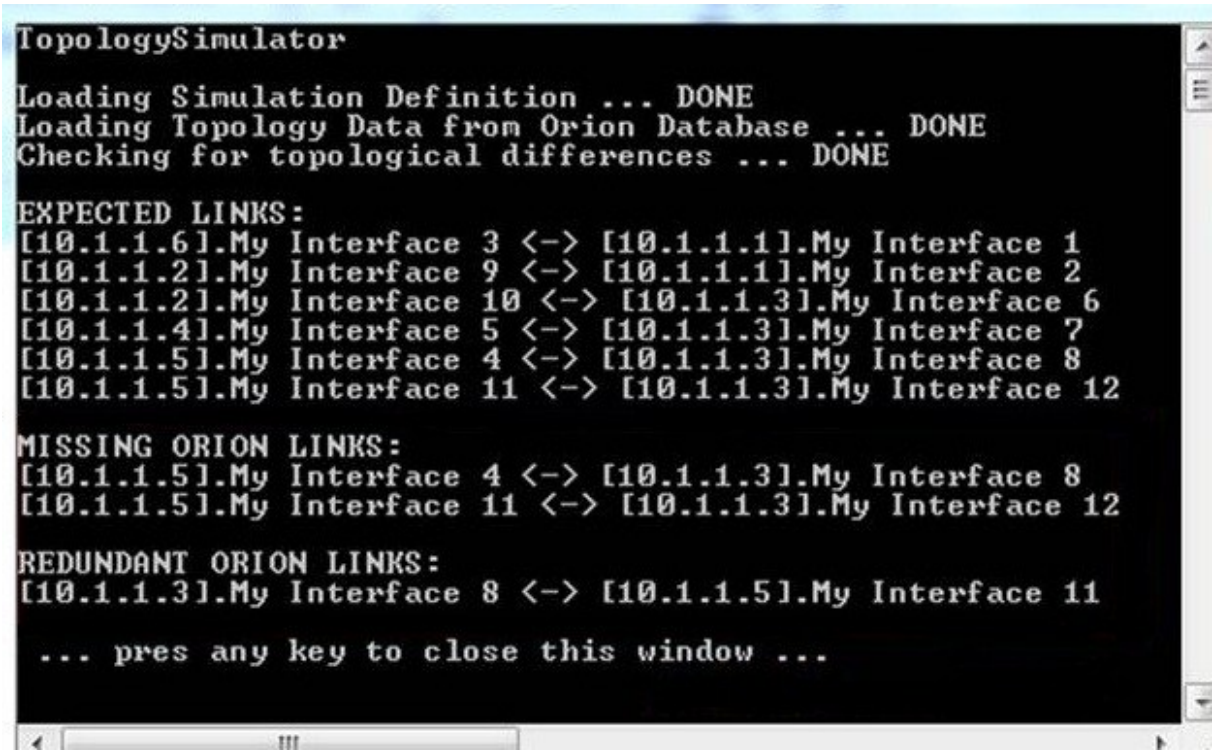
Poslední karta, kterou oddíl s informacemi obsahuje, je karta Links. Zde je seznam všech spojů pro vybrané dvojice uzlů (Obrázek 4.6). Kliknutím vybereme rozhraní zdrojového uzlu (Link Source) a následně rozhraní cílového uzlu (Link Destination). Tlačítko s oranžovou šipkou vpravo přidá dvojici rozhraní jako spoj do seznamu. Spojení se zobrazí také graficky na mapě v podobě čáry



Obrázek 4.6: Oddíl s informacemi ohledně spojů – karta Links

mezi odpovídající dvojící uzlů. V případě že bude spojů mezi uzly více, na mapce zůstane stále pouze jedna čára. Kliknutím na tuto čáru se zobrazí spoje mezi odpovídající dvojící uzlů.

Součástí hlavního okna jsou také ovládací prvky, které najdeme v levé horní části aplikace. Jsou to funkce pro vytvoření nového zadání, uložení a načtení stavu nastavení do souboru xml. Dále ovládací prvek pro spuštění emulace, který po kliknutí spustí procesy pro vygenerování MIB definic, spustí skript pro nastavení IP adres síťovému adaptéru, nakonfiguruje parametry souboru config.xml pro SNMP Responder a spustí SNMP Responder. Poslední tlačítko (Compare Results) porovná data spočtené topologie z databáze Orionu s daty v naší aplikaci a zobrazí výsledky (Obrázek 4.7).



Obrázek 4.7: Okno s výsledkem porovnání

Vidíme zde seznam očekávaných spojů, tj. spojů které jsme si v aplikaci nadefinovali. Dále seznam spojů, které v Orionu chybí. Nakonec také redundantní spoje, které Orion zjistil chybně.

5 Srovnání a praktické výsledky

V této kapitole popíšu, jakým způsobem jsem testoval naimplementovanou aplikaci a shrnu zde výsledky.

5.1 Testovací prostředí

Jelikož se ve firmě SolarWinds používá virtualizované prostředí, využil jsem k testování licencovaného produktu VMware Workstation, který mi firma poskytla. Na virtuálním stroji byl nainstalovaný operační systém Windows Server 2008 R2 ve 64-bitové verzi a další potřebné aplikace:

- SQL Server 2008 R2
- Web Server IIS 7.5
- .NET Framework 4.0

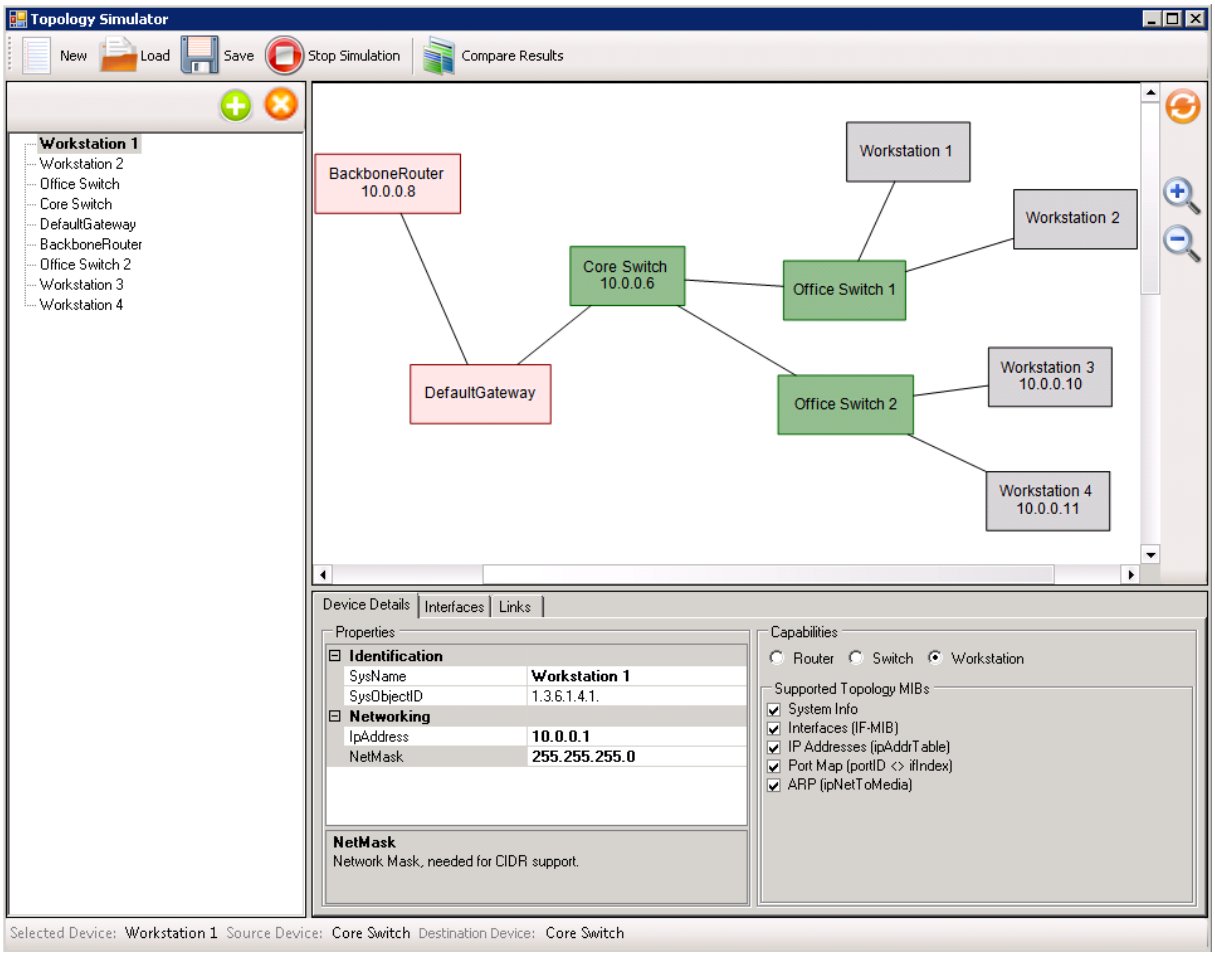
Samozřejmě zde byl také nainstalován produkt Network Performance Monitor aktuální verze 10.5, na kterém budeme mojí aplikaci testovat. Stroj běžel s procesorem Intel Xeon 2,4 GHz (2 jádra), 4GB RAM operační pamětí a 500GB pevným diskem.

5.2 Nasazení a výsledky testování

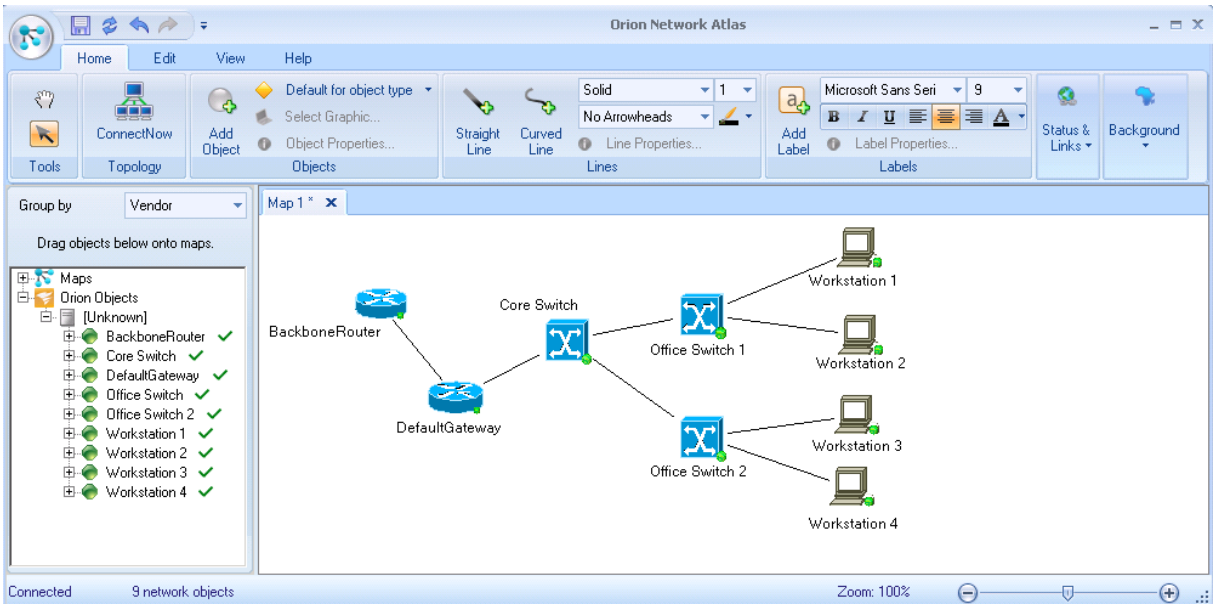
Dříve než naši aplikaci nasadíme, je třeba zkontrolovat, zdali v operačním systému náhodou není nainstalovaná a spuštěná služba SNMP (SNMP Service). V případě, že v systému tato služba je, musíme ji buď vypnout nebo změnit výchozí port (161) v konfiguraci SNMP Responderu na některý jiný. Je to z toho důvodu, že na portu 161 již poslouchá služba SNMP systému Windows a tím pádem by SNMP Responder tento obsazený port už nemohl využít. Emulace by nám pak nefungovala.

Dále je třeba otevřít konfigurační soubor app.config a v něm správně nastavit název síťového adaptéru, na kterém budou jednotlivá emulovaná zařízení odpovídat (resp. na jeho IP adresách). Výchozí název adaptéru je zde „Local Area Connection“, takže pokud budeme chtít testovat např. v české verzi operačního systému Windows, bude třeba tento řetězec změnit. V konfiguračním souboru je obsažen také řetězec pro připojení do databáze produktu Orion. Z této databáze budeme brát data pro porovnání výsledků.

Testování pomocí naší aplikace potvrdilo, že zpětné rekonstruování topologie sítě je složitý úkol. Produkt Orion Network Performance Monitor na mých datech fungoval dobře (Obrázek 5.1, Obrázek 5.2), nicméně ukázalo se, že i zde jsou určité rezervy. Šlo například o omezení v podobě nemožnosti identifikovat větší počet linek mezi uzly. Celkově byla firma s výsledky spokojena a v současné době se připravují další topologické scénáře.



Obrázek 5.1: Emulovaná síť prostřednictvím aplikace TopologySimulator



Obrázek 5.2: Zjištěná síť programem Orion Network Atlas

6 Závěr

Cílem této diplomové práce bylo vytvoření aplikace pro emulaci síťových prvků propojených podle zadané síťové topologie komunikující přes protokol SNMP v prostředí jednoho počítače. Aplikace byla navrhnuta na míru potřebám softwarového produktu Orion Network Performance Monitor firmy SolarWinds, která byla také jejím zadavatelem.

K emulaci zařízení bylo možné využít již existujícího nástroje SNMP Responder, který firma SolarWinds vlastní. Nad tímto jsem vytvořil svou aplikaci implementující grafické uživatelské rozhraní, pomocí kterého je uživatel schopen nastavit potřebné parametry jednotlivých aktivních prvků, jejich propojení, vygenerovat odpovídající SNMP data pro SNMP Responder a spustit emulaci sítě. Výsledek je ten, že máme k dispozici program, který se umí postarat nejen o nastavení parametrů a vygenerování dat, ale také o samotnou emulaci sítě a všeho ostatního, ačkoliv k tomu využívá dalšího nástroje. Uživatel už jen přidá emulované zařízení do produktu Orion Network Performance Monitor a počká na výsledky, které moje aplikace rovněž dokáže porovnat s původním zadáním a vyhodnotit správnost. Cíle tedy byly splněny.

Přínos mé aplikace spočívá v možnosti zařazení mé aplikace do testovacího procesu firmy. Nasazení je možné v rámci manuálních či automatizovaných testů. V rámci automatizace by bylo možné připravit několik předdefinovaných map a testovat tak pravidelně zdali např. dílčí změna v kódu produktu Orion nezpůsobila problém s algoritmem pro kalkulaci topologie.

Budoucí vylepšení a vývoj se může ubírat několika směry. Co se týče generování dat, mohli bychom zde zavést „šum“. To znamená negenerovat kompletní data - rozbít předpoklad, že každý s každým komunikoval a určitým způsobem některá data vynechat. Umožnit např. nadefinování seznamu uzlů a linek, které spolu nekomunikovali a vyřadit jej tak z dat, která budou generována. S tímto lze pak lépe testovat robustnost algoritmu pro výpočet topologie. Také by šel napodobit např. princip STP, kdy se přepínače mezi sebou dohodnou, že budou mezi sebou používat pro komunikaci pouze některé linky a určí tímto komunikační trasy. Reálně bývají v síti také různé výpadky či jiné konfigurační problémy a veškerá data mnohdy k dispozici stejně nejsou. Dalším možným směrem je optimalizace výpočtu prohledávání sítě, kdy se pro každý uzel prohledává opakovaně to samé okolí a výpočetní složitost je tak vcelku vysoká. Zde by bylo možné vymyslet pokročilejší logiku prohledání s pamatováním si, že jednou nalezené uzly jsou již propojené a tím pádem nezjišťovat tuto skutečnost několikrát dokola. Posledním možným směrem je implementace topologických dat na úrovni 3. vrstvy ISO/OSI modelu. Tj. přidat směrovací informace a ty poskytovat v generovaných datech.

Použitá literatura

[1] BOUŠKA, Petr. SNMP - Simple Network Management Protocol. *Samuraj-cz* [online]. 2006 [cit. 2012-03-29]. Dostupné z: http://www.samuraj-cz.com/clanek/snmp-simple-network-management-protocol/

[2] Vznik a principy SNMP. *Svět sítí: Správa počítačových sítí* [online]. 2000 [cit. 2012-03-29]. Dostupné z: http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&clanekID=29

[3] SNMP protokol a jeho využití. *Hw.cz* [online]. 2003 [cit. 2012-03-29]. Dostupné z: http://www.hw.cz/Produkty/ART957-SNMP-protokol-a-jeho-vyuziti.html

[4] SNMP správa zařízení - Úvod a terminologie. *AdminXP.cz* [online]. 2010 [cit. 2012-03-29]. Dostupné z: http://www.adminxp.cz/networking/index.php?aid=21

[5] MAURO, D. R., SCHMIDT, K. J. Essential SNMP. Sebastopol: O'Reilly Media, 2005. ISBN 978-0-596-00840-6.

[6] RFC 1157. A Simple Network Management Protocol (SNMP) [online]. CASE, J., FEDOR, M., SCHOFFSTALL, M., DAVIN, J. May 1990 [cit. 2012-03-29]. Dostupné z: http://www.ietf.org/rfc/rfc/1157.txt

[7] KOZIEROK, C. M. The TCP/IP guide: a comprehensive, illustrated Internet protocols reference. San Francisco: No Starch Press, 2005, on p. 1080. ISBN 9781593270476.

[8] Cisco Discovery Protocol. In Cisco Systems: Technology support [online]. San Jose: Cisco Systems [cit. 2013-02-15] Dostupné z: http://www.cisco.com/en/US/tech/tk648/tk362/tk100/tsd_technology_support_sub-protocol_home.html

[9] Using Link Layer Discovery Protocol in Multivendor Networks. In Cisco Systems: Cisco IOS and NX-OS Software [online]. San Jose: Cisco Systems, 2007 [cit. 2013-03-02]. Dostupné z: http://cisco.com/en/US/docs/ios/cether/configuration/guide/ce_lldp-med.html

[10] PERLMAN, R. Interconnections. Boston: Addison-Wesley, 2000. ISBN 0-201-63448-1-

[11] STALLINGS, William. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. 3, illustrated. USA : Addison-Wesley, 1999. 619 s. ISBN 9780201485349.

[12] WALSH, Larry. SNMP MIB handbook: essential guide to MIB development, use and diagnosis. illustrated. USA : Wyndham Press, 2008. 408 s. ISBN 9780981492209.

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha A: Adresářová struktura přiloženého DVD

/Aplikace	Obsahuje spustitelnou aplikaci.
/Projekt	Kompletní Visual Studio 2012 projekt celé aplikace. Obsahuje zdrojové kódy.
/Text	Text diplomové práce ve formátu .docx a .pdf